Smart city

Fachhochschule Dortmund
University of Applied Sciences and Arts

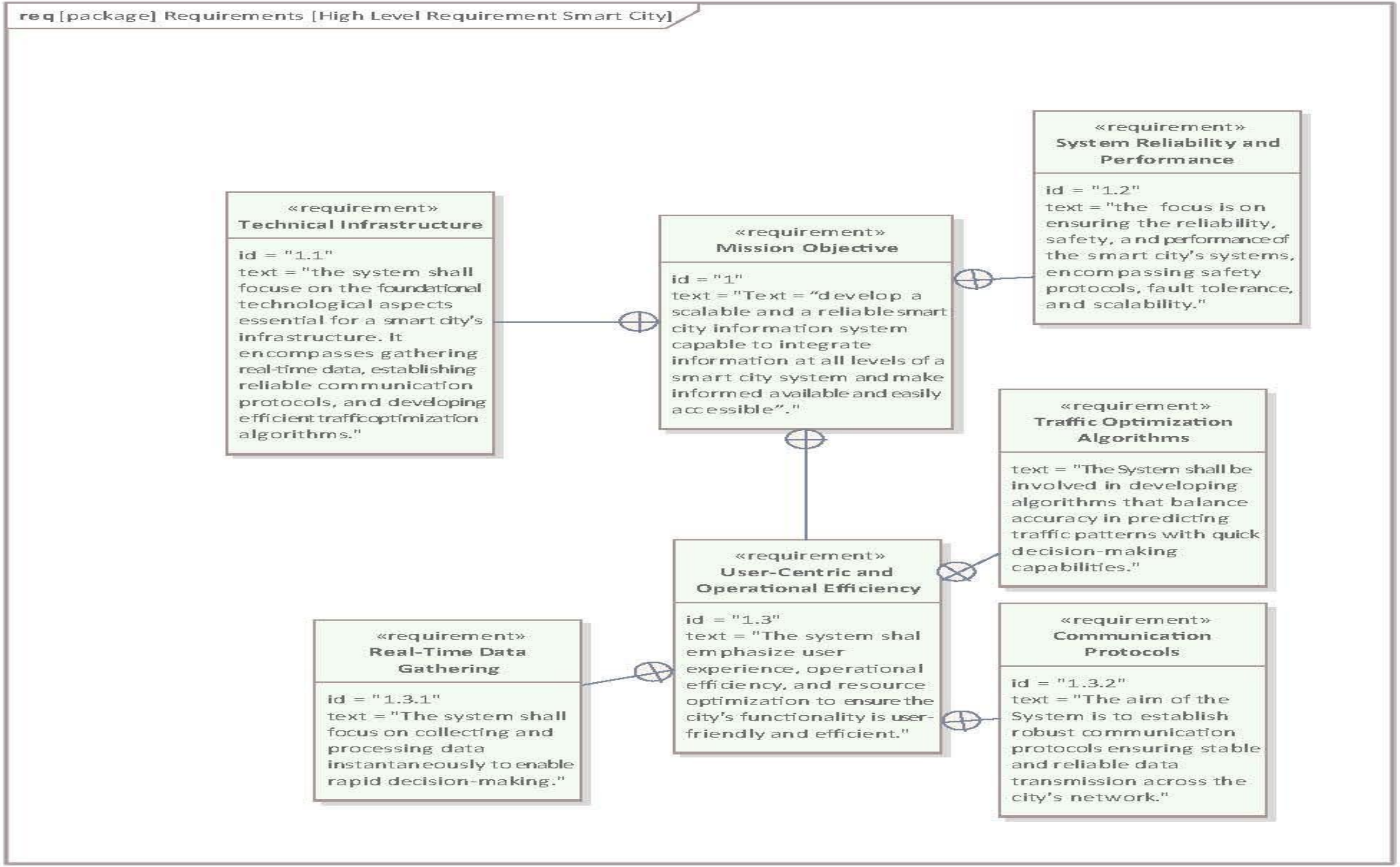
https://omrania.com/insights/smart-cities-need-culture-and-community-not-just-data/

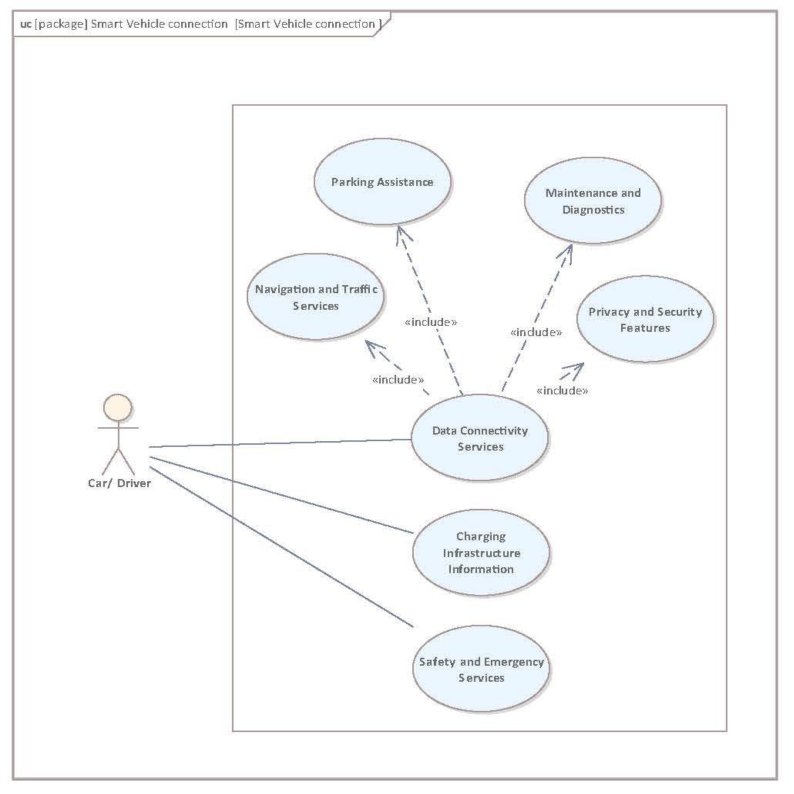
INTRODUCTION

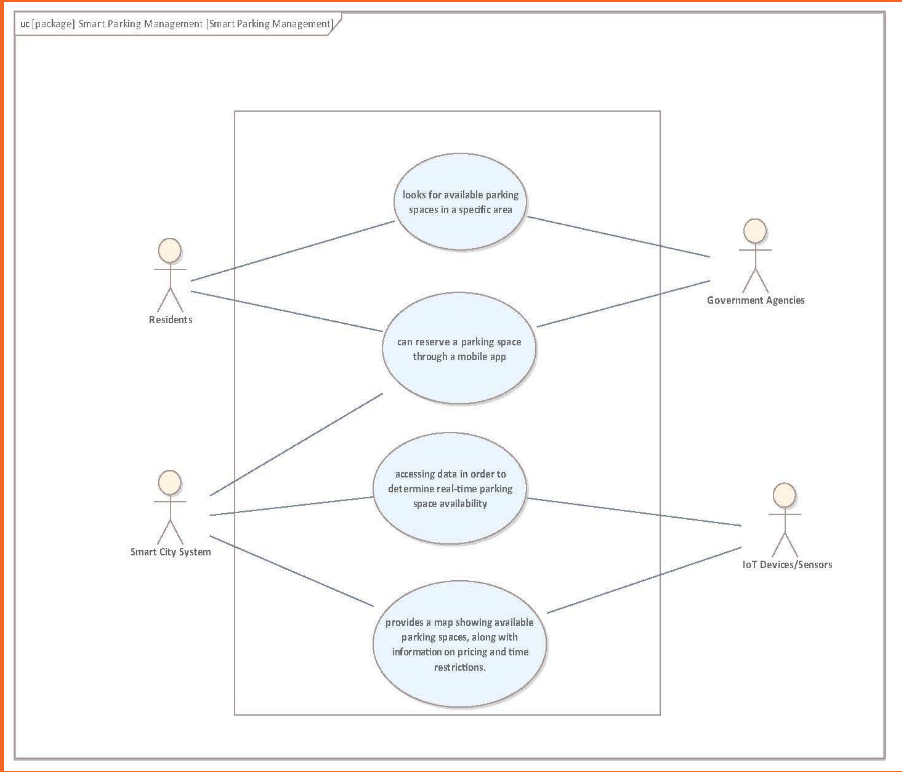
This document explores and presents our Smart City project, a team effort to transform urban living through innovative technology. We're working together to redefine how cities function by introducing smart solutions.

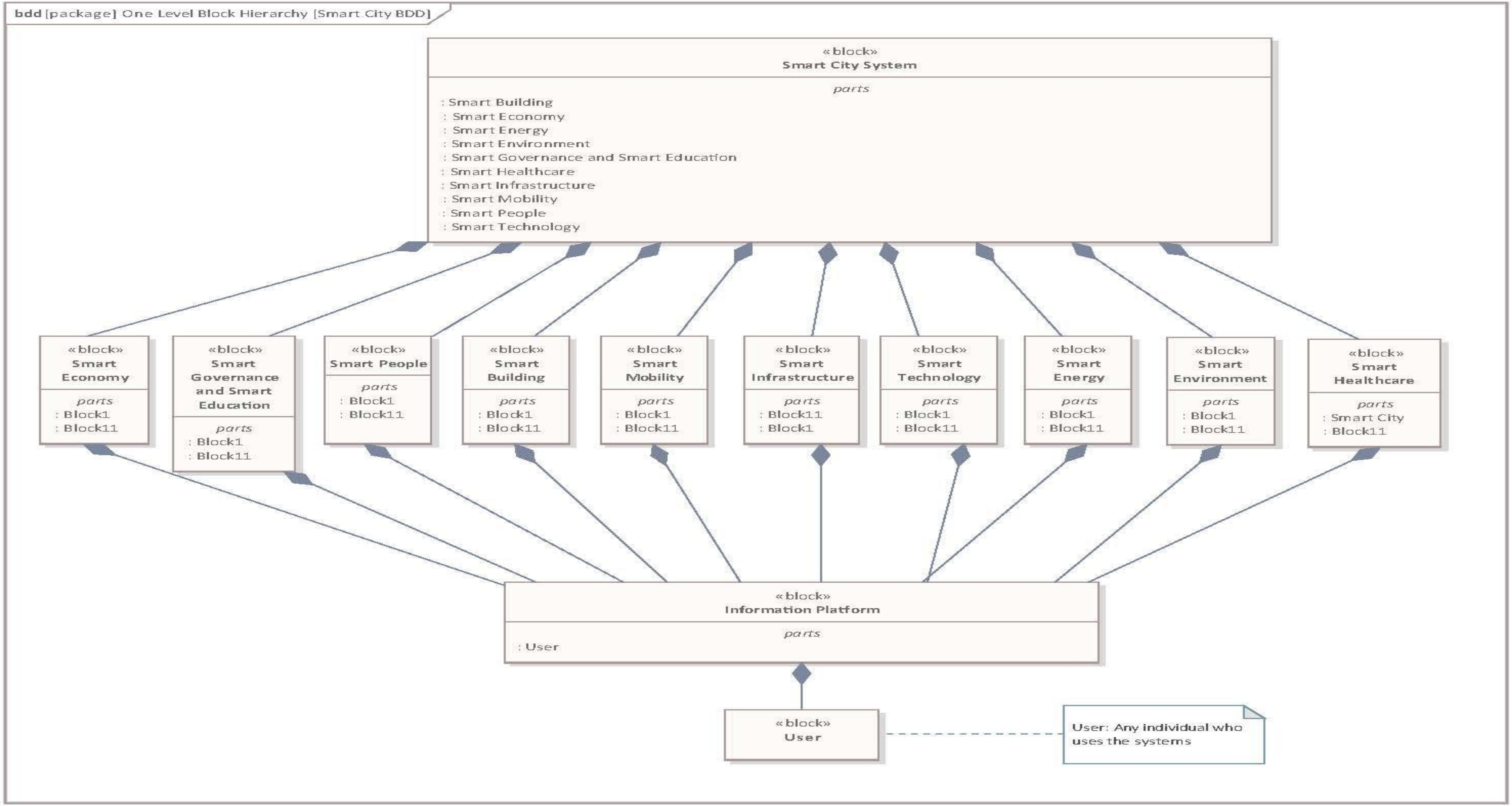
Key achievements

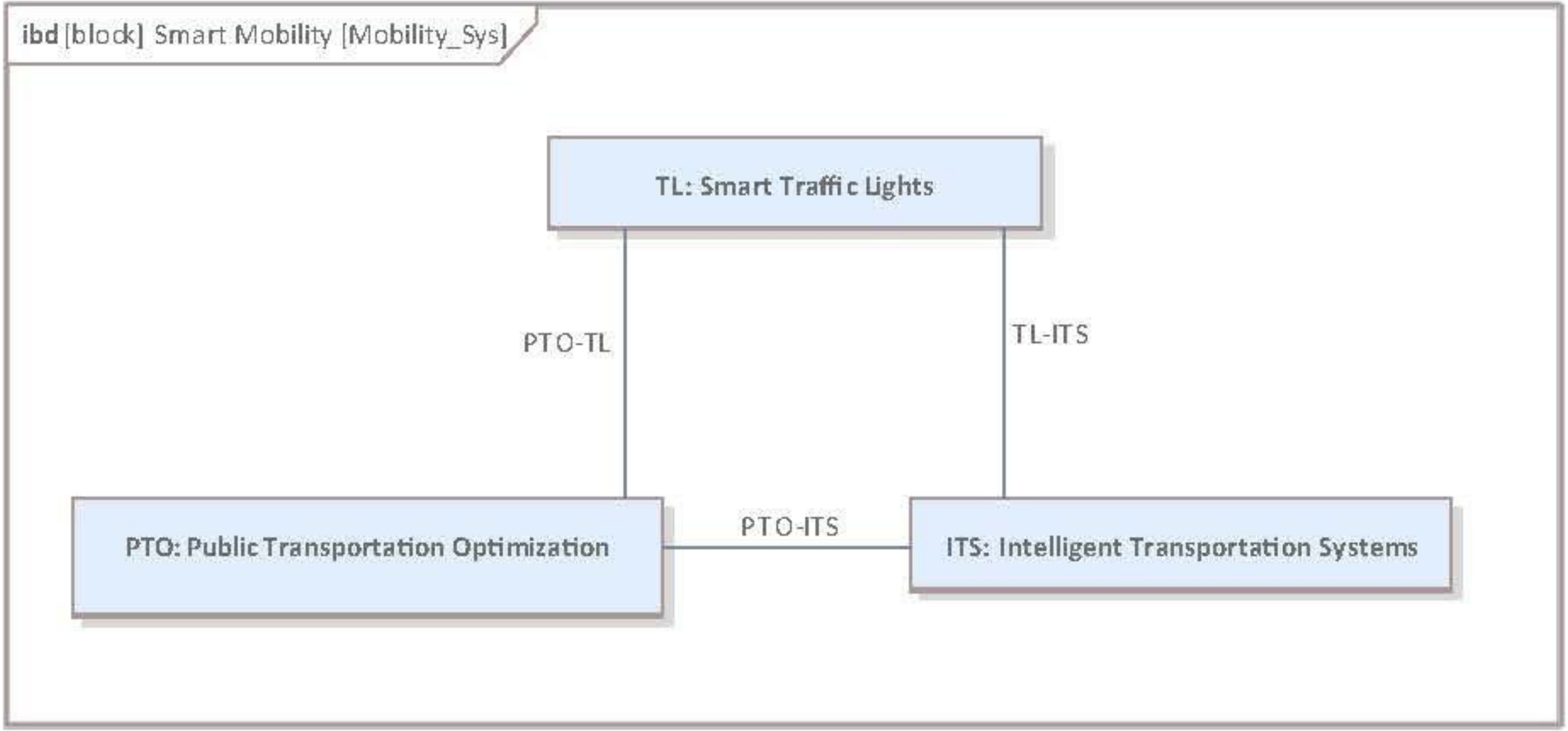
- Energy Consumption
- Fast mobility
- Low Traffic jam
- EMergency scenario

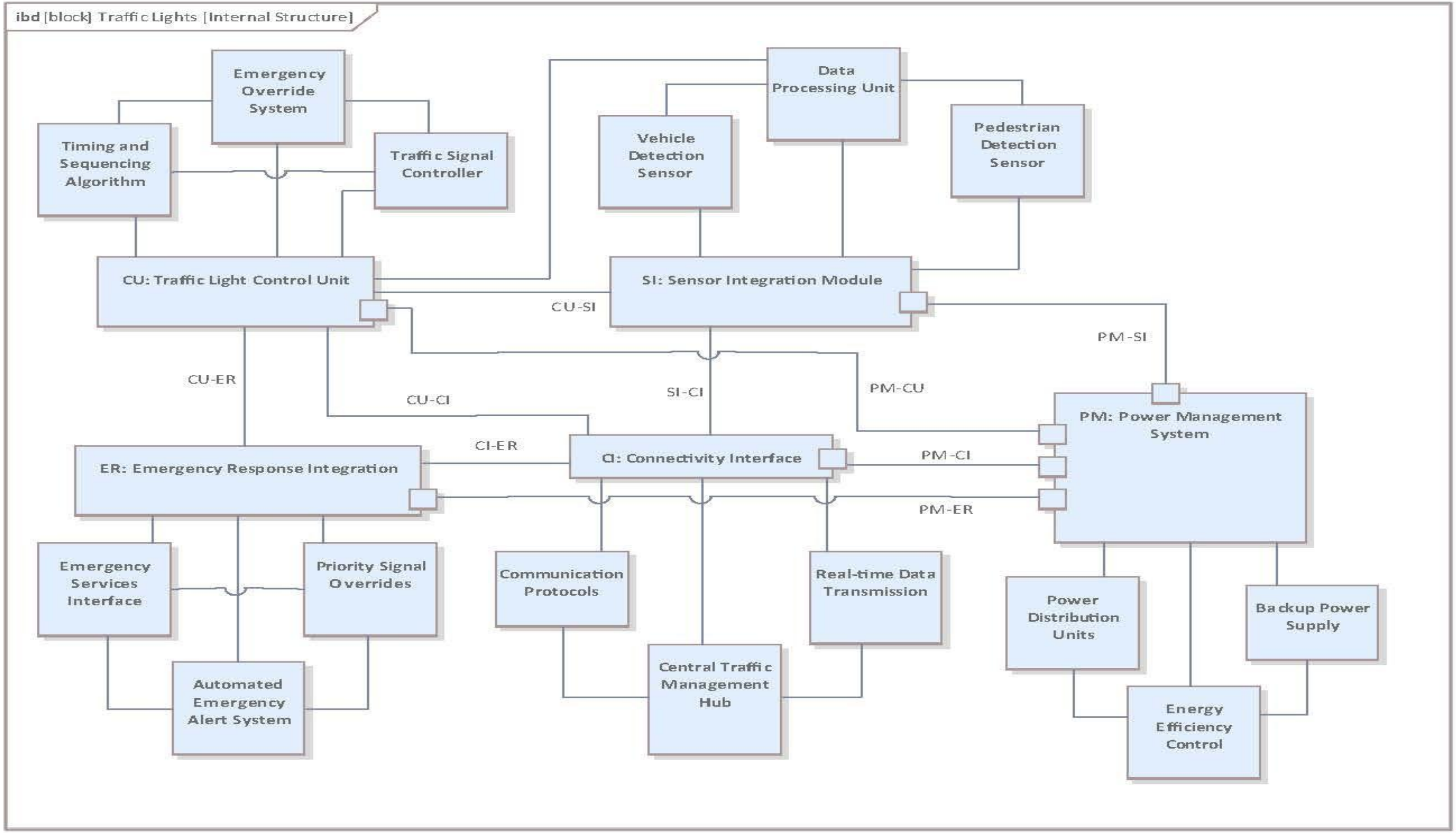

req [package] Requirements [High Level Requirement Smart City]
«requirement»
System Reliability and Performance
id = "1.2"
text = "the focus is on ensuring the reliability, safety, and performance of the smart city's systems, encompassing safety protocols, fault tolerance, and scalability."
«requirement»
Technical Infrastructure
id = "1.1"
text = "the system shall focuse on the foundational technological aspects essential for a smart city's infrastructure. It encompasses gathering real-time data, establishing reliable communication protocols, and developing efficient trafficoptimization algorithms."
«requirement»
Mission Objective
id = "1"
text = "Text = "develop a scalable and a reliable smart city information system capable to integrate information at all levels of a smart city system and make informed available and easily accessible"."
«requirement»
Traffic Optimization Algorithms
text = "The System shall be involved in developing algorithms that balance accuracy in predicting traffic patterns with quick decision-making capabilities."
«requirement»
User-Centric and Operational Efficiency
id = "1.3"
text = "The system shal emphasize user experience, operational efficiency, and resource optimization to ensure the city's functionality is user-friendly and efficient."
«requirement»
Real-Time Data Gathering
id = "1.3.1"
text = "The system shall focus on collecting and processing data instantaneously to enable rapid decision-making."
«requirement»
Communication Protocols
id = "1.3.2"
text = "The aim of the System is to establish robust communication protocols ensuring stable and reliable data transmission across the city's network."


uc [package] Smart Vehicle connection [Smart Vehicle connection]
Parking Assistance
Maintenance and Diagnostics
Navigation and Traffic Services
Privacy and Security Features
«include»
«include»
«include»
«include»
Data Connectivity Services
Car/ Driver
Charging Infrastructure Information
Safety and Emergency Services


uc [package] Smart Parking Management [Smart Parking Management]
looks for available parking spaces in a specific area
Residents
Government Agencies
can reserve a parking space through a mobile app
accessing data in order to determine real-time parking space availability
Smart City System
IoT Devices/Sensors
provides a map showing available parking spaces, along with information on pricing and time restrictions.

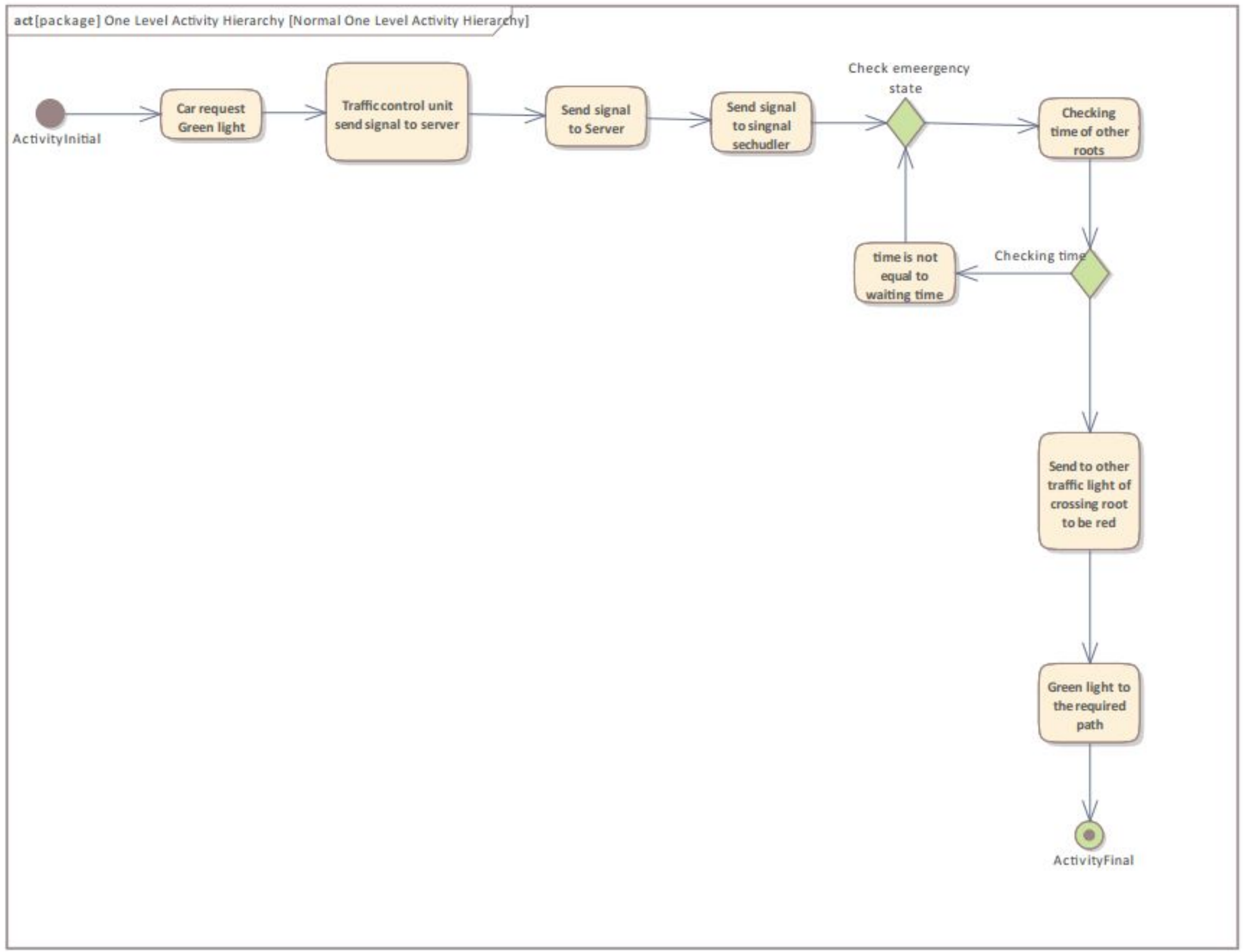

bdd [package] One Level Block Hierarchy [Smart City BDD]
«block»
Smart City System
parts
: Smart Building
: Smart Economy
: Smart Energy
: Smart Environment
: Smart Governance and Smart Education
: Smart Healthcare
: Smart Infrastructure
: Smart Mobility
: Smart People
: Smart Technology
«block»
Smart Economy
parts
: Block1
: Block11
«block»
Smart Governance and Smart Education
parts
: Block1
: Block11
«block»
Smart People
parts
: Block1
: Block11
«block»
Smart Building
parts
: Block1
: Block11
«block»
Smart Mobility
parts
: Block1
: Block11
«block»
Smart Infrastructure
parts
: Block11
: Block1
«block»
Smart Technology
parts
: Block1
: Block11
«block»
Smart Energy
parts
: Block1
: Block11
«block»
Smart Environment
parts
: Block1
: Block11
«block»
Smart Healthcare
parts
: Smart City
: Block11
«block»
Information Platform
parts
: User
«block»
User
User: Any individual who uses the systems


ibd [block] Smart Mobility [Mobility_Sys]
TL: Smart Traffic Lights
PTO-TL
TL-ITS
PTO-ITS
PTO: Public Transportation Optimization
ITS: Intelligent Transportation Systems

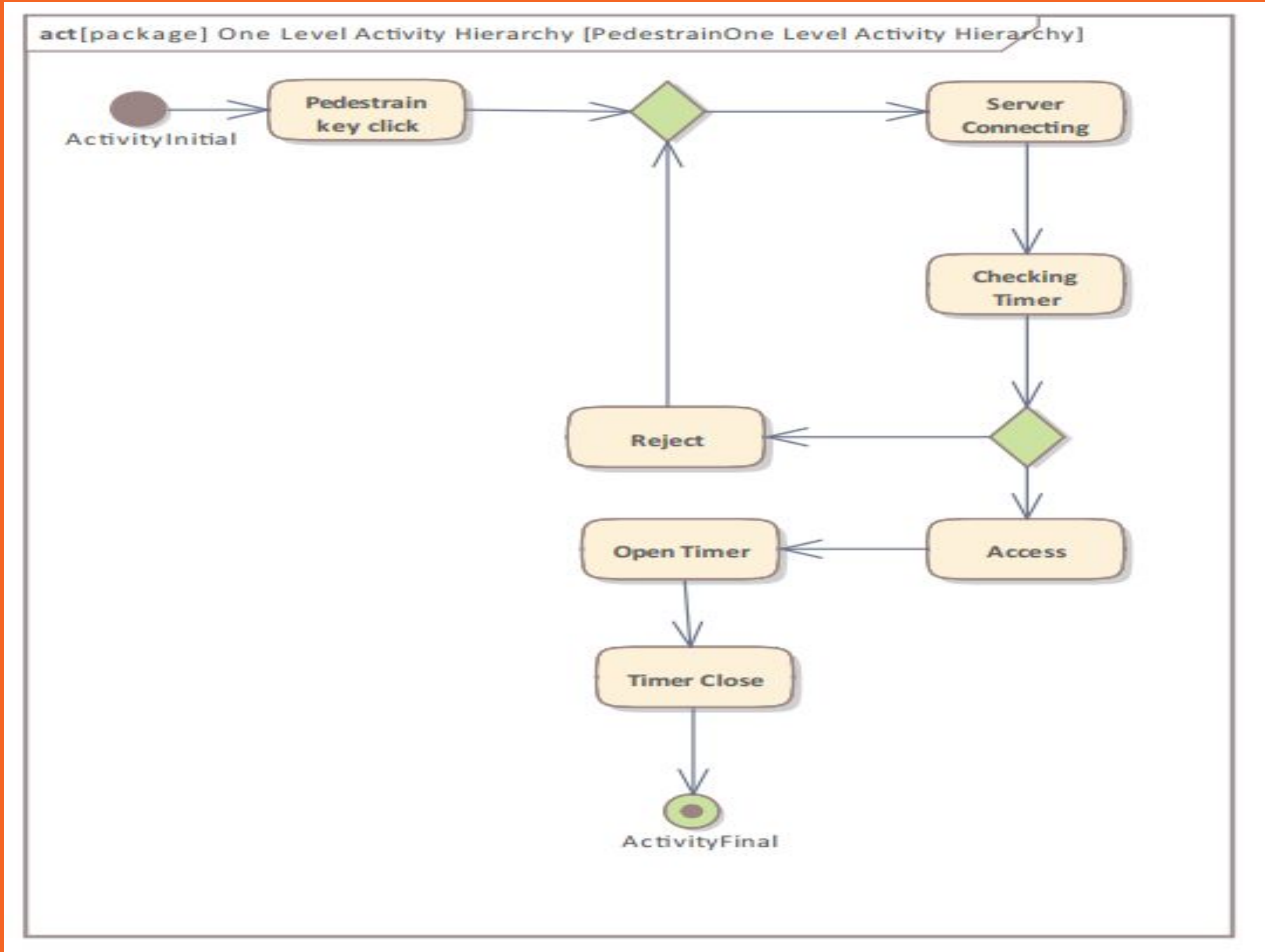

ibd [block] Traffic Lights [Internal Structure]
Emergency Override System
Timing and Sequencing Algorithm
Traffic Signal Controller
Data Processing Unit
Vehicle Detection Sensor
Pedestrian Detection Sensor
CU: Traffic Light Control Unit
SI: Sensor Integration Module
CU-SI
PM-SI
CU-ER
CU-CI
SI-CI
PM-CU
PM: Power Management System
CI-ER
CI: Connectivity Interface
PM-CI
ER: Emergency Response Integration
PM-ER
Emergency Services Interface
Priority Signal Overrides
Communication Protocols
Real-time Data Transmission
Power Distribution Units
Backup Power Supply
Automated Emergency Alert System
Central Traffic Management Hub
Energy Efficiency Control

Activity Diagram

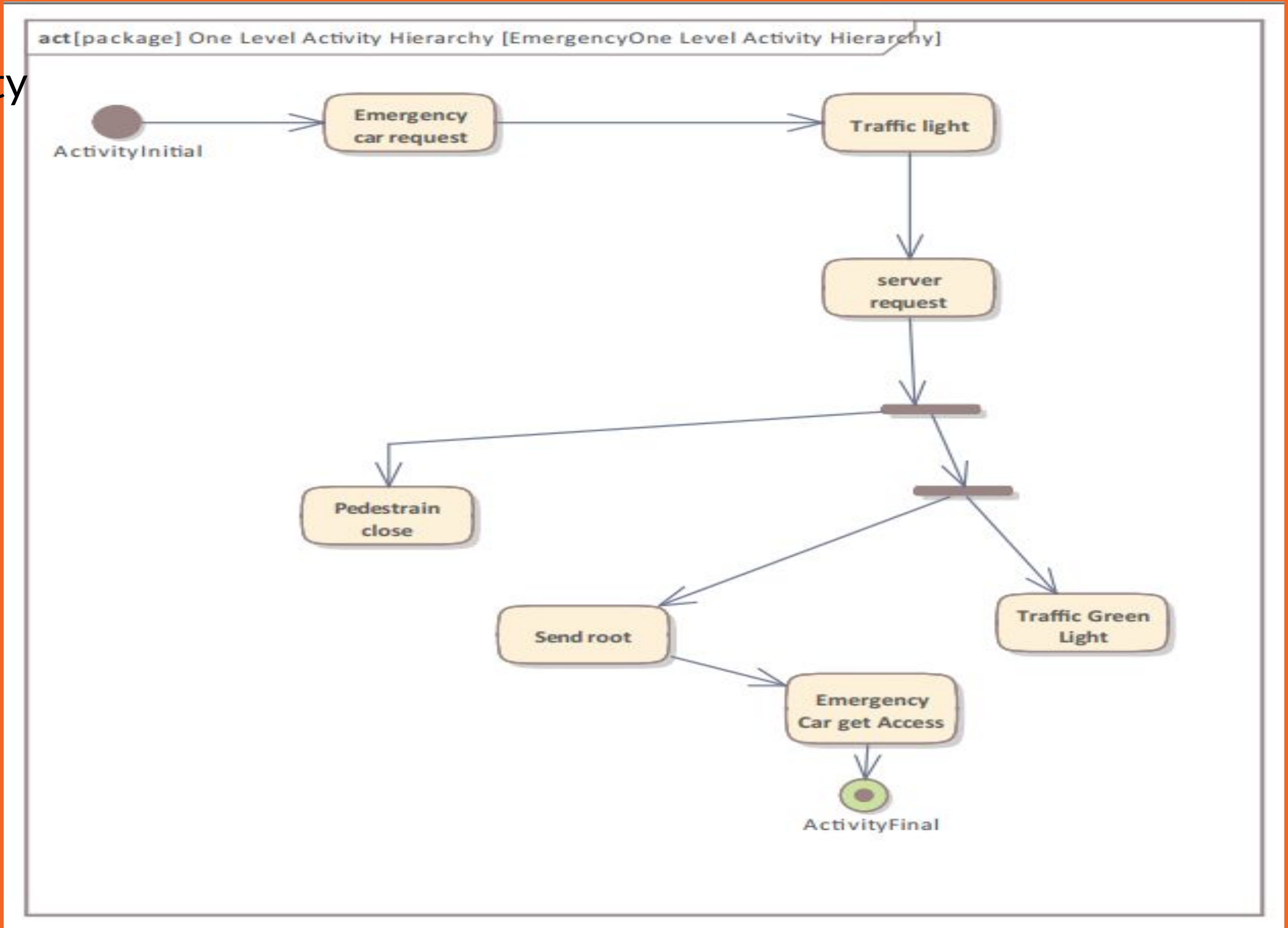
Normal case


act [package] One Level Activity Hierarchy [Normal One Level Activity Hierarchy]
ActivityInitial
Car request Green light
Traffic control unit send signal to server
Send signal to Server
Send signal to singnal sechudler
Check emeergency state
Checking time of other roots
Checking time
time is not equal to waiting time
Send to other traffic light of crossing root to be red
Green light to the required path
ActivityFinal

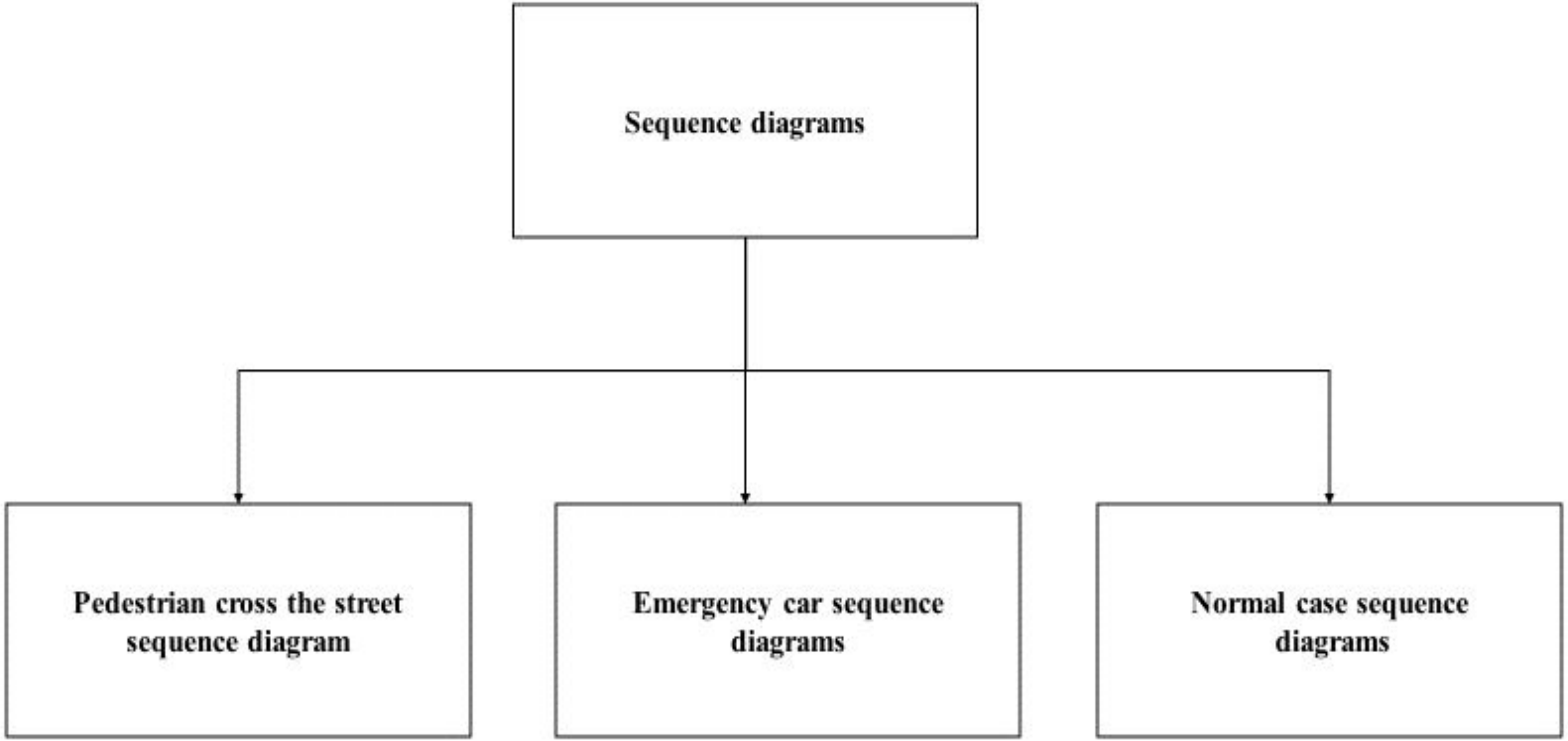
Pedestrian case



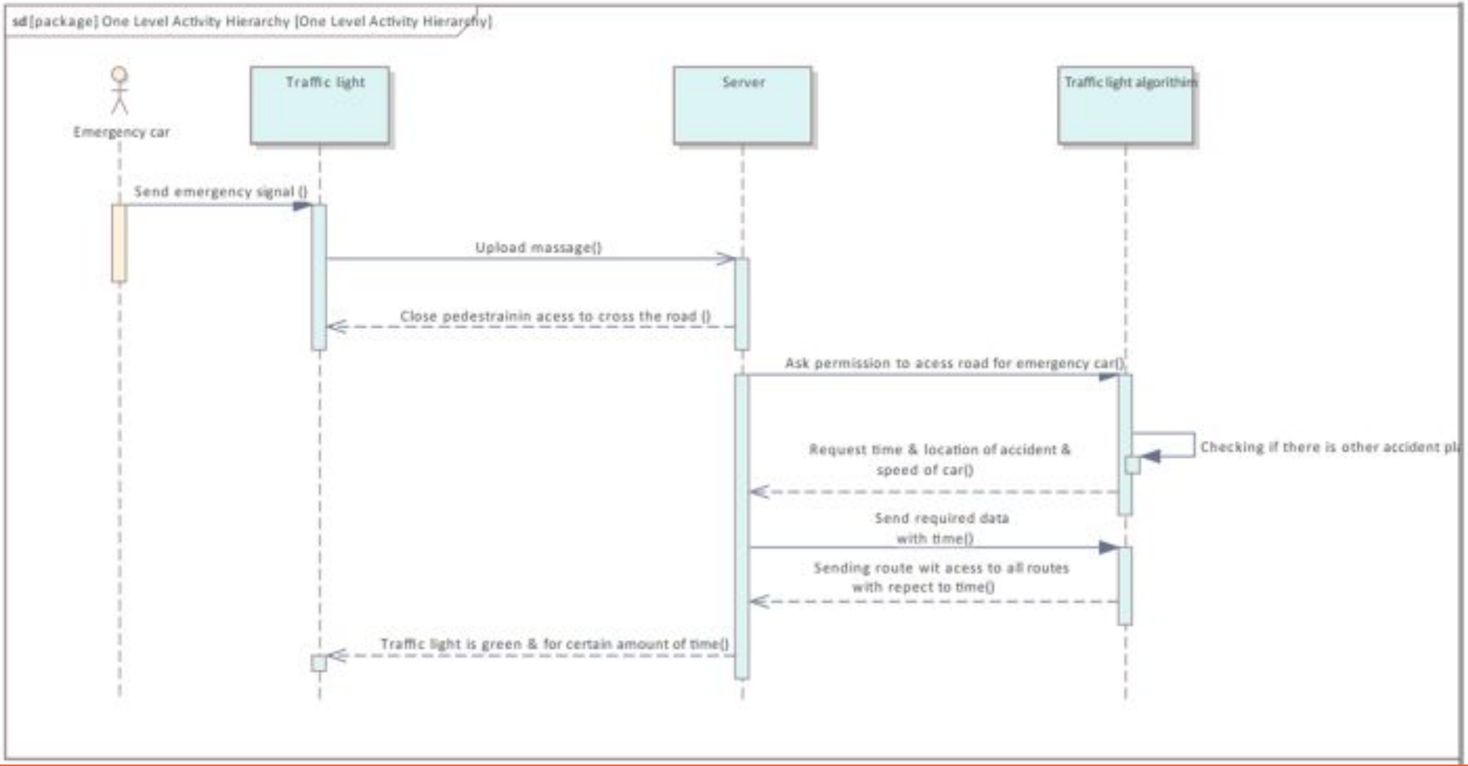
Emergency activity



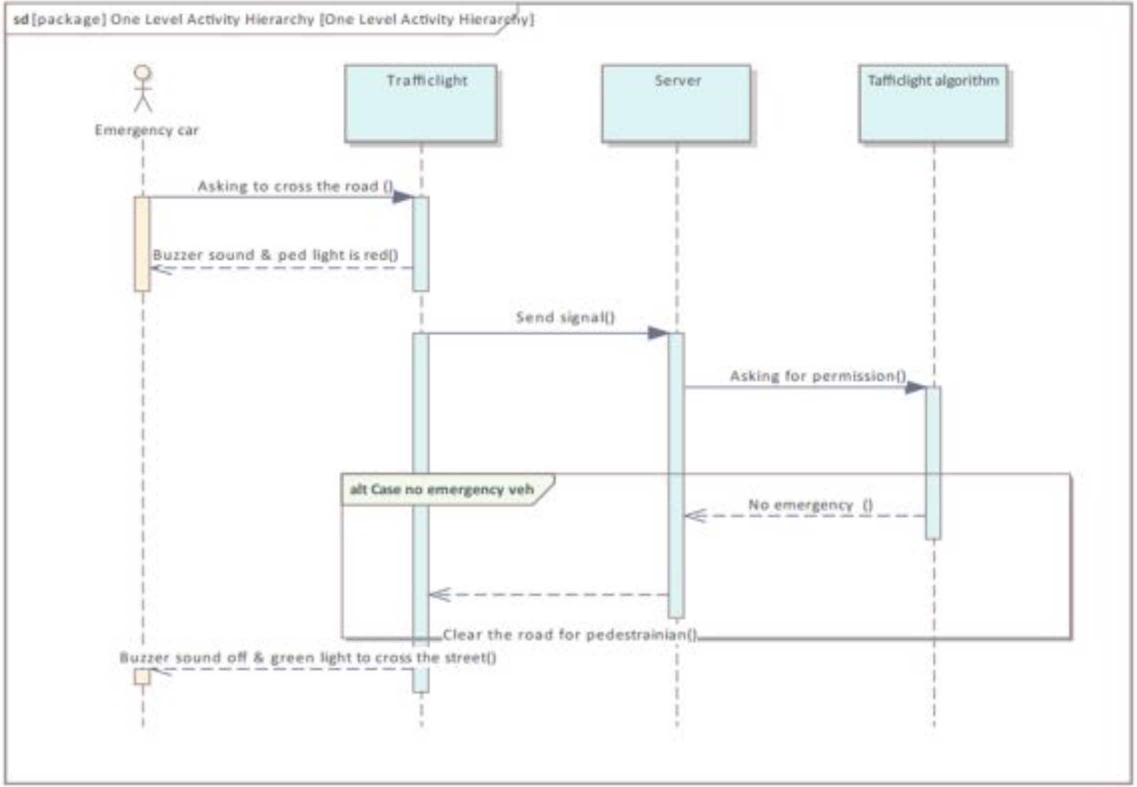
Sequence diagram


Sequence diagrams
Pedestrian cross the street sequence diagram
Emergency car sequence diagrams
Normal case sequence diagrams

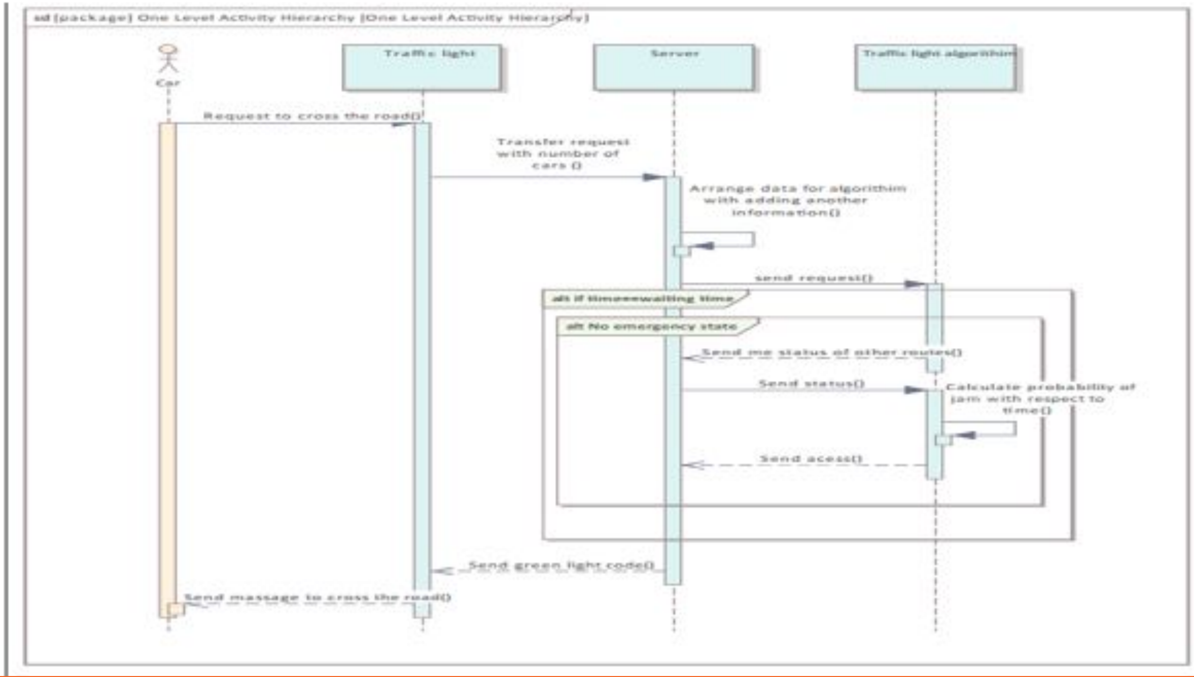
Emergency car crossing the street sequence diagram



Pedestrian crossing the street sequence diagram

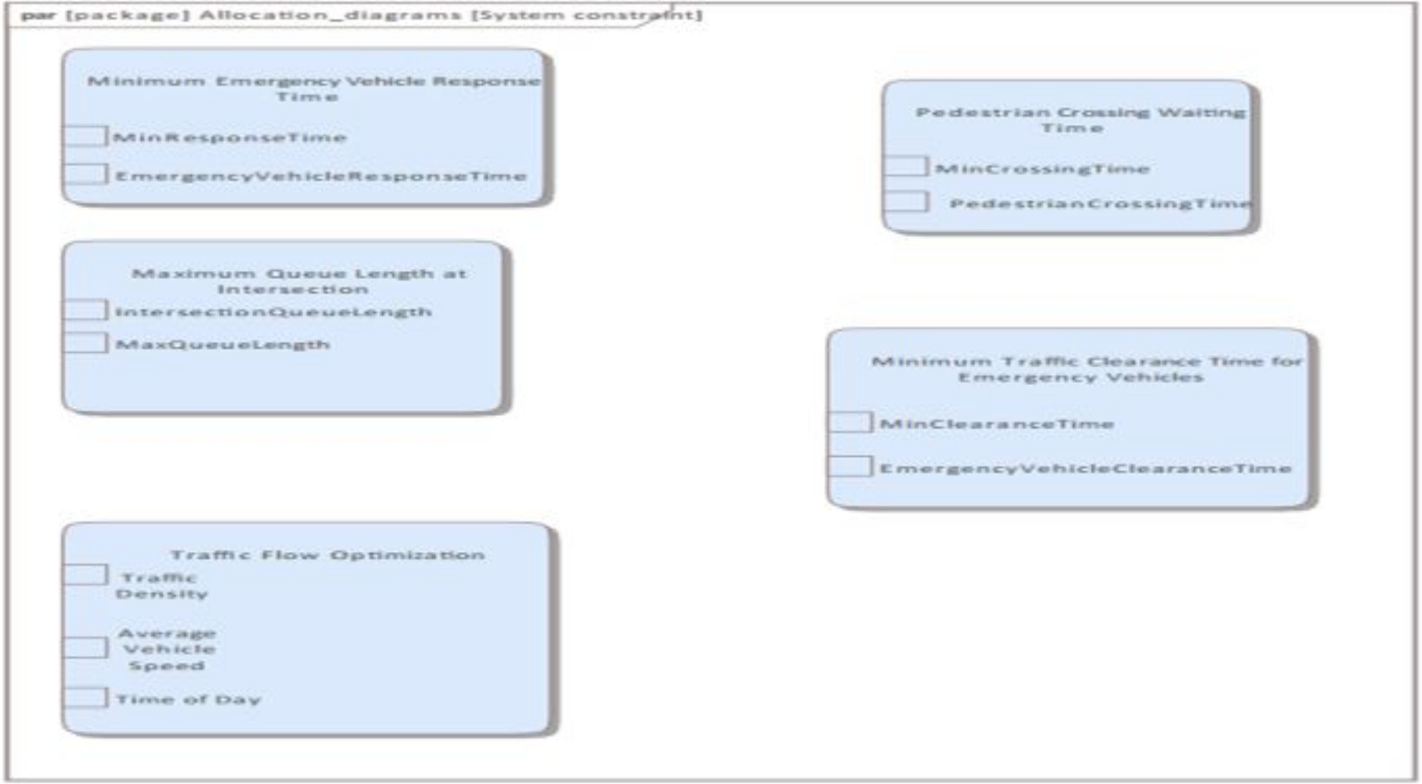


Normal Situation sequence diagram

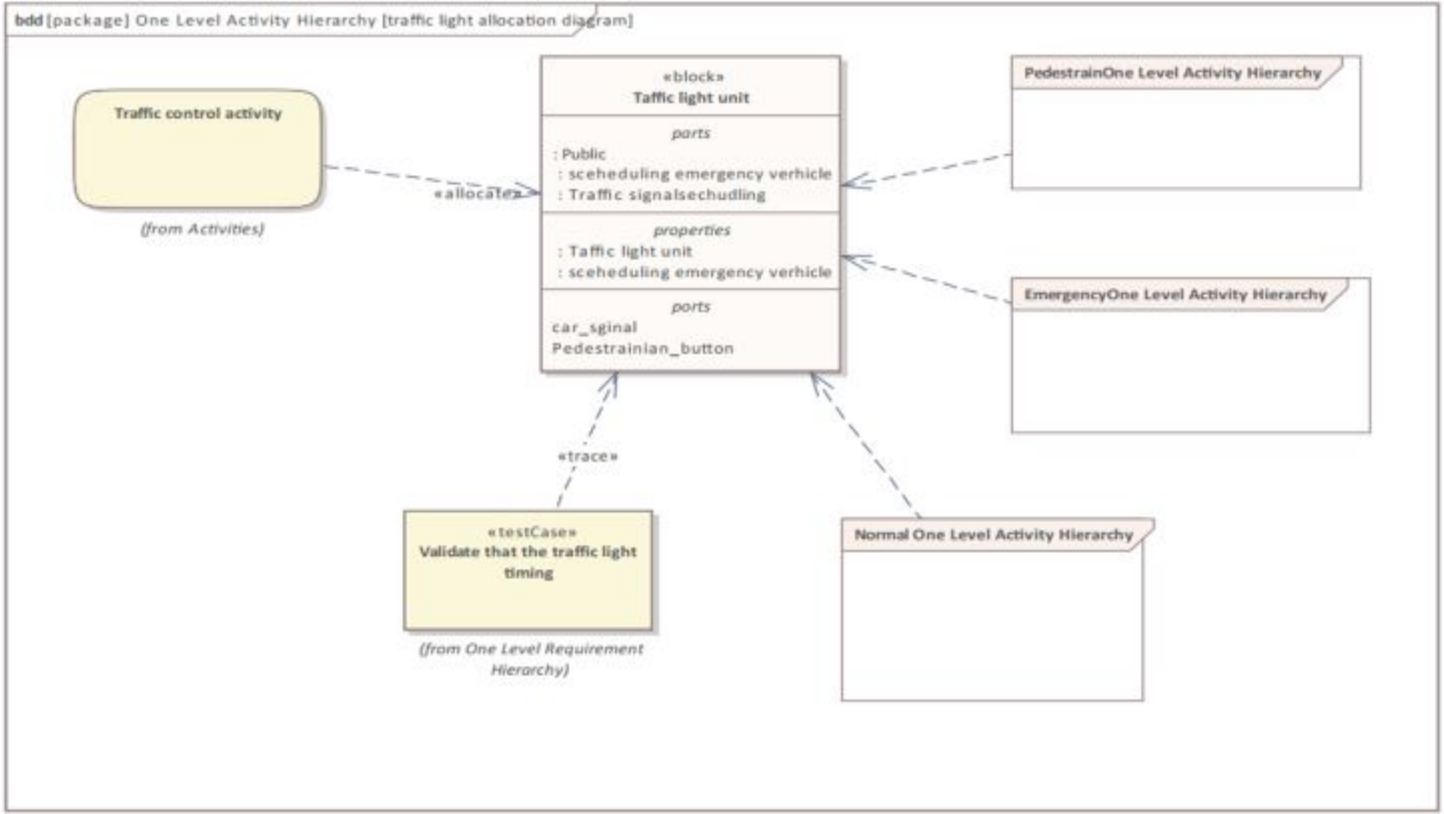


Design Phase

➢ Constraint Diagram



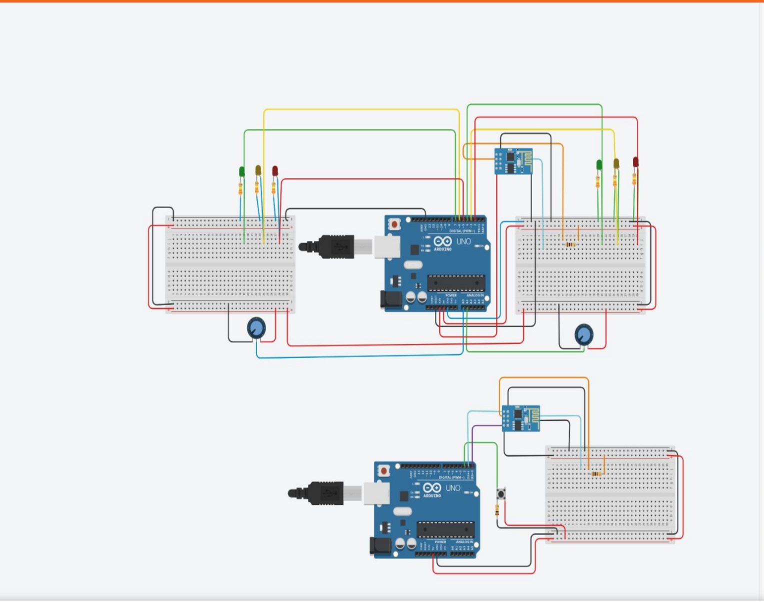
➢ Allocation diagram



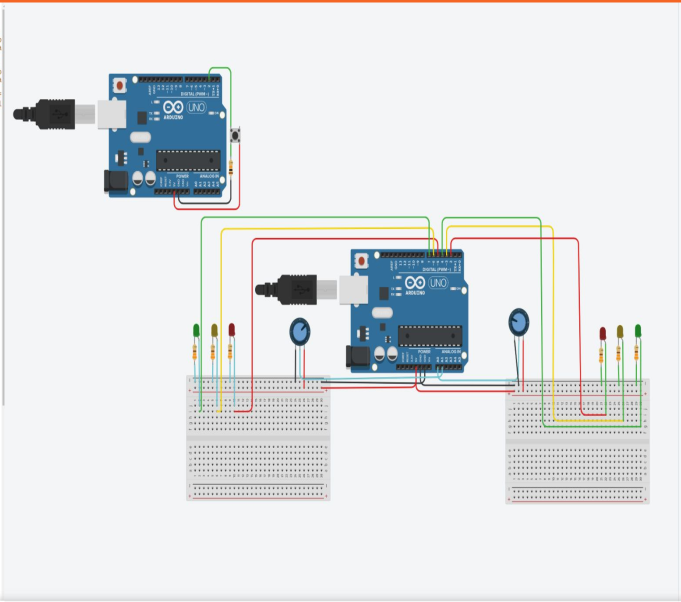
Implementation and Testing

- Using two bluetooth modules
- Using USART
- Normal circuit
- Unit Testing using google cpp framwork

These are the two systems

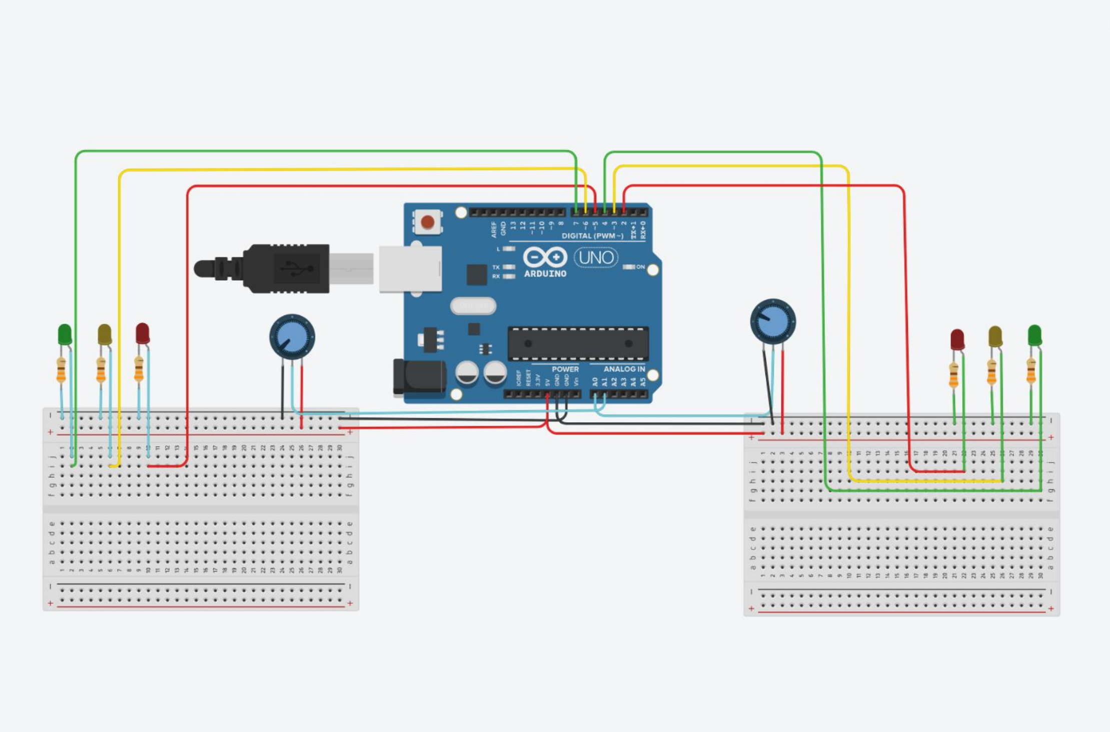


```
const int potPinRoad1 = A0;  // Potentiometer for Road 1
const int potPinRoad2 = A1;  // Potentiometer for Road 2

const int redPinRoad1 = 2;    // Digital pin for red LED on Road
const int yellowPinRoad1 = 3; // Digital pin for yellow LED on Ro
const int greenPinRoad1 = 4;  // Digital pin for green LED on Roa

const int redPinRoad2 = 5;    // Digital pin for red LED on Road
const int yellowPinRoad2 = 6; // Digital pin for yellow LED on Ro
const int greenPinRoad2 = 7;  // Digital pin for green LED on Roa

volatile boolean messageReceived = false;  // Flag to indicate if
boolean inAmbulanceMode = false;  // Flag to indicate if currentl

void setup() {
  pinMode(redPinRoad1, OUTPUT);
  pinMode(yellowPinRoad1, OUTPUT);
  pinMode(greenPinRoad1, OUTPUT);

  pinMode(redPinRoad2, OUTPUT);
  pinMode(yellowPinRoad2, OUTPUT);
  pinMode(greenPinRoad2, OUTPUT);

  Serial.begin(9600);
}

void loop() {
  if (Serial.available() > 0) {
    Serial.println("balabizo");
    // Handle the received message
    handleReceivedMessage();
  }

  // Your other tasks go here

  int potValueRoad1 = analogRead(potPinRoad1);
  int potValueRoad2 = analogRead(potPinRoad2);

  int durationRoad1 = map(potValueRoad1, 0, 1023, 5000, 15000);
  int durationRoad2 = map(potValueRoad2, 0, 1023, 5000, 15000);

  Serial.print("Road 1 Duration: ");
  Serial.println(durationRoad1);
  Serial.print("Road 2 Duration: ");
  Serial.println(durationRoad2);

  if (potValueRoad1 > potValueRoad2) {
    if (!inAmbulanceMode) {
      digitalWrite(redPinRoad2, HIGH);
      digitalWrite(yellowPinRoad2, LOW);
      digitalWrite(greenPinRoad2, LOW);
      delay(1000);

      digitalWrite(redPinRoad1, LOW);
      digitalWrite(yellowPinRoad1, HIGH);
      digitalWrite(greenPinRoad1, LOW);
      delay(2000);

      digitalWrite(redPinRoad1, LOW);
      digitalWrite(yellowPinRoad1, LOW);
      digitalWrite(greenPinRoad1, HIGH);
      delay(durationRoad1);
    }
  } else {
    if (!inAmbulanceMode) {
      digitalWrite(redPinRoad1, HIGH);
      digitalWrite(yellowPinRoad1, LOW);
      digitalWrite(greenPinRoad1, LOW);
      delay(1000);

      digitalWrite(redPinRoad2, LOW);
      digitalWrite(yellowPinRoad2, HIGH);
      digitalWrite(greenPinRoad2, LOW);
      delay(2000);
```



```
const int buttonPin = 2;  // Digital pin for the button
char ambulanceMessage = '1';  // Special message to indicate an am

void setup() {
  pinMode(buttonPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  if (digitalRead(buttonPin) == HIGH) {
    // Button is pressed, send the ambulance message
    Serial.print(ambulanceMessage);
    delay(1000);  // Add a small delay to avoid multiple messages
  }
  // Your other tasks go here
}
```

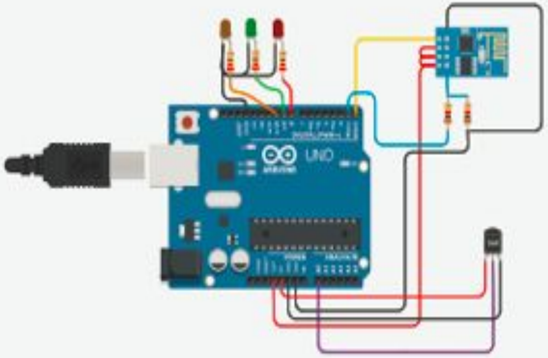
Final System

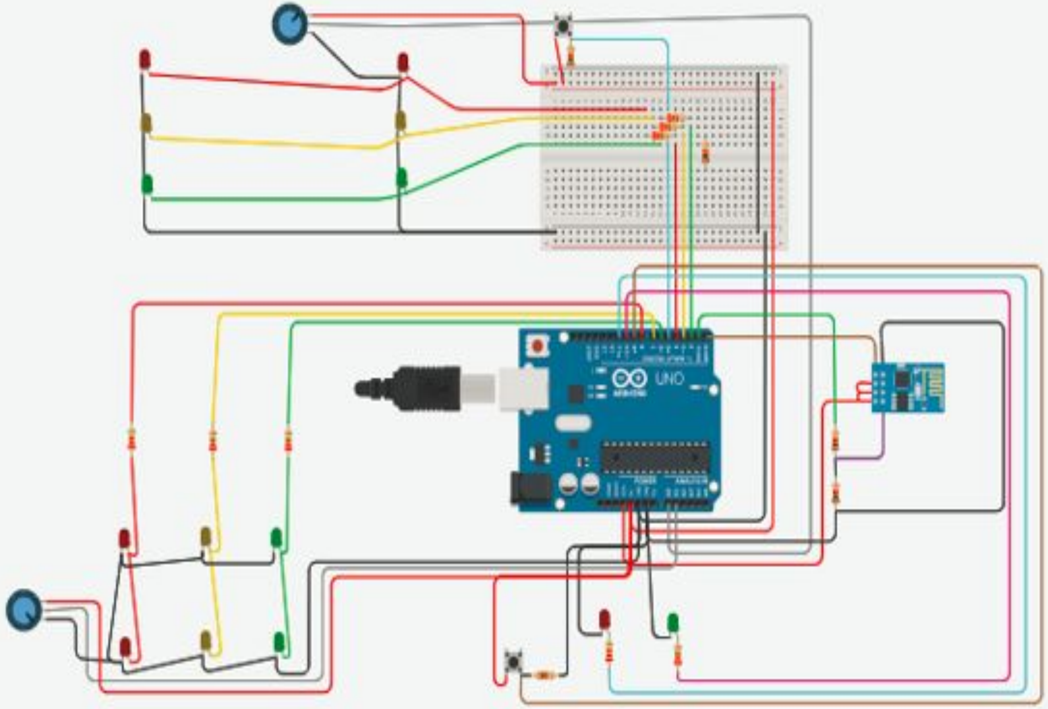


Text | 1 (Arduino Uno R3)

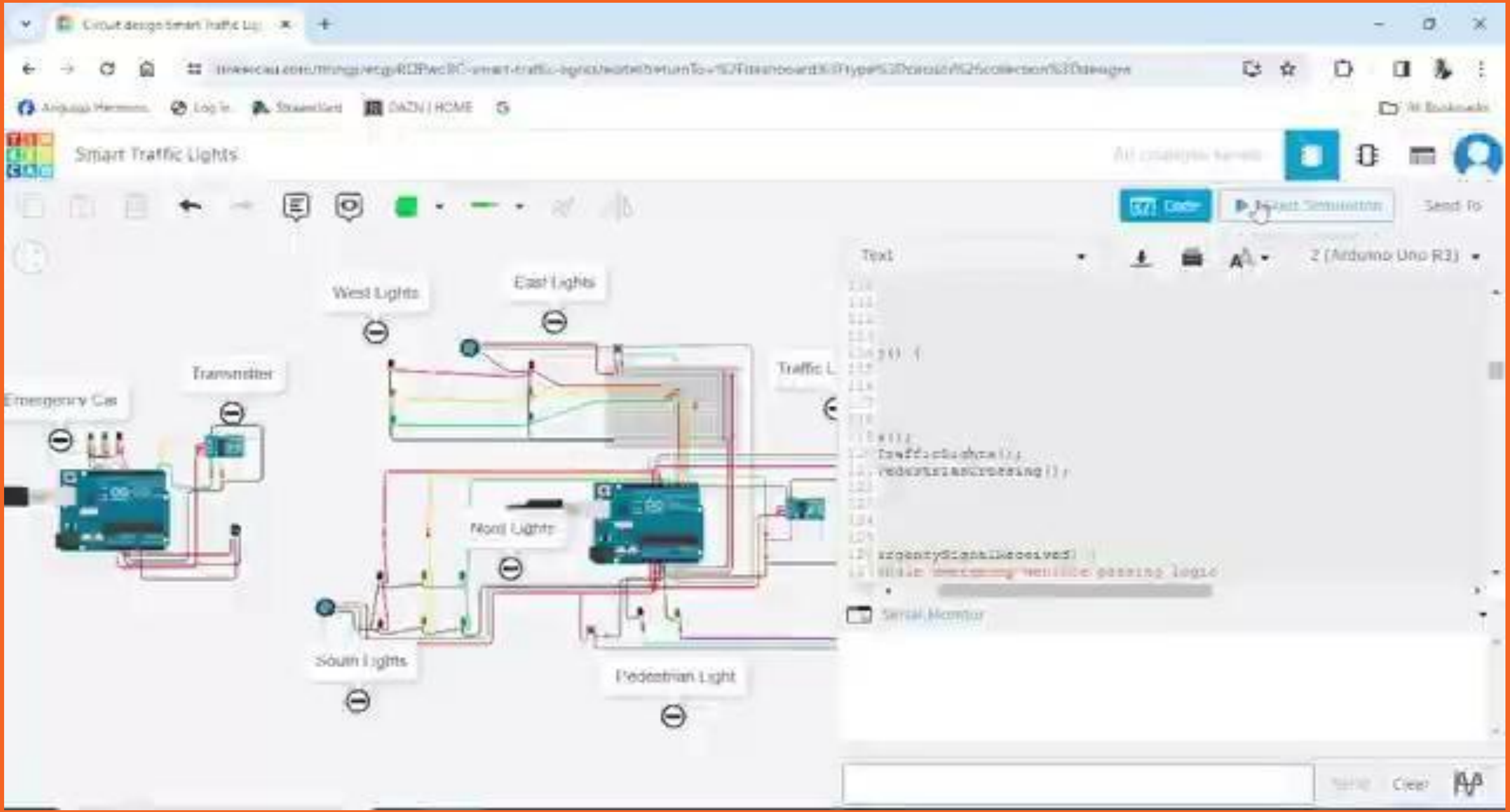
```
const int potPinRoad1 = A0;  // Potentiometer for Road 1
const int potPinRoad2 = A1;  // Potentiometer for Road 2

const int redPinRoad1 = 2;    // Digital pin for red LED on Road 1
const int yellowPinRoad1 = 3; // Digital pin for yellow LED on Roac
const int greenPinRoad1 = 4;  // Digital pin for green LED on Road

const int redPinRoad2 = 5;    // Digital pin for red LED on Road 2
const int yellowPinRoad2 = 6; // Digital pin for yellow LED on Roac
const int greenPinRoad2 = 7;  // Digital pin for green LED on Road

void setup() {
  pinMode(redPinRoad1, OUTPUT);
  pinMode(yellowPinRoad1, OUTPUT);
  pinMode(greenPinRoad1, OUTPUT);

  pinMode(redPinRoad2, OUTPUT);
  pinMode(yellowPinRoad2, OUTPUT);
  pinMode(greenPinRoad2, OUTPUT);

  Serial.begin(9600);
}

void loop() {
  int potValueRoad1 = analogRead(potPinRoad1);
  int potValueRoad2 = analogRead(potPinRoad2);

  int durationRoad1 = map(potValueRoad1, 0, 1023, 5000, 15000);  //
  int durationRoad2 = map(potValueRoad2, 0, 1023, 5000, 15000);  //

  Serial.print("Road 1 Duration: ");
  Serial.println(durationRoad1);
  Serial.print("Road 2 Duration: ");
  Serial.println(durationRoad2);

  if (potValueRoad1 > potValueRoad2) {
    digitalWrite(redPinRoad2, HIGH);
    digitalWrite(yellowPinRoad2, LOW);
    digitalWrite(greenPinRoad2, LOW);
    delay(1000);

    digitalWrite(redPinRoad1, LOW);
    digitalWrite(yellowPinRoad1, HIGH);
    digitalWrite(greenPinRoad1, LOW);
    delay(2000);

    digitalWrite(redPinRoad1, LOW);
    digitalWrite(yellowPinRoad1, LOW);
    digitalWrite(greenPinRoad1, HIGH);
    delay(durationRoad1);
  } else {
    digitalWrite(redPinRoad1, HIGH);
    digitalWrite(yellowPinRoad1, LOW);
    digitalWrite(greenPinRoad1, LOW);
    delay(1000);

    digitalWrite(redPinRoad2, LOW);
    digitalWrite(yellowPinRoad2, HIGH);
    digitalWrite(greenPinRoad2, LOW);
    delay(2000);

    digitalWrite(redPinRoad2, LOW);
    digitalWrite(yellowPinRoad2, LOW);
    digitalWrite(greenPinRoad2, HIGH);
    delay(durationRoad2);
  }


  digitalWrite(redPinRoad1, LOW);
  digitalWrite(yellowPinRoad1, LOW);
  digitalWrite(greenPinRoad1, LOW);
```

Emergency Vehicle





Traffic Lights System


Smart Traffic Lights
West Lights
East Lights
Transmitter
Emergency Car
North Lights
South Lights
Pedestrian Light
Text
Serial Monitor
Clear

Unit test cases

```
TEST_F(TrafficControllerTest, MapWrapperFunction) {
    // Test mapWrapper function
    int result = controller.mapWrapper(512, 0, 1023, 5000, 15000);
    EXPECT_EQ(result, 10004);
}

TEST_F(TrafficControllerTest, LoopFunctionWithZeroDuration) {
    // Test loop function with zero duration
    int potValueRoad1 = 0;
    int potValueRoad2 = 0;
    controller.loop(potValueRoad1, potValueRoad2);
    // You can add assertions based on the expected behavior of the loop function
}

TEST_F(TrafficControllerTest, LoopFunctionWithEqualPotValues) {
    // Test loop function with equal pot values
    int potValueRoad1 = 512;
    int potValueRoad2 = 512;
    controller.loop(potValueRoad1, potValueRoad2);
    // You can add assertions based on the expected behavior of the loop function
}

TEST_F(TrafficControllerTest, LoopFunctionWithPot1GreaterThanPot2) {
    // Test loop function with pot1 greater than pot2
    int potValueRoad1 = 800;
    int potValueRoad2 = 400;
    controller.loop(potValueRoad1, potValueRoad2);
    // You can add assertions based on the expected behavior of the loop function
}
```

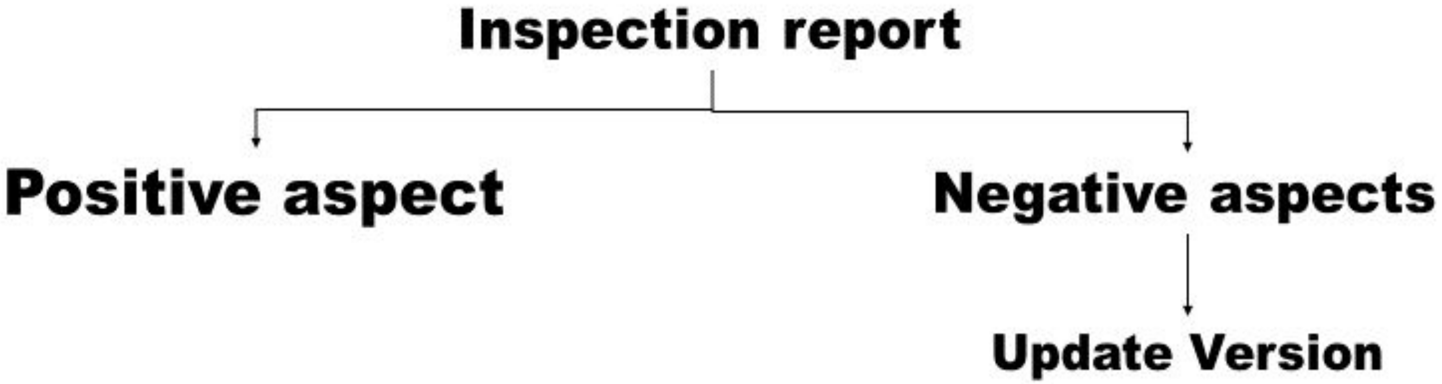
```cpp
TrafficController::TrafficController() {
    // Constructor implementation, if needed
}

void TrafficController::setup() {
    std::cout << "Simulating pinMode" << std::endl;
}

void TrafficController::loop(int potValueRoad1, int potValueRoad2) {
    int durationRoad1 = map(potValueRoad1, 0, 1023, 5000, 15000);
    int durationRoad2 = map(potValueRoad2, 0, 1023, 5000, 15000);

    std::cout << "Road 1 Duration: " << durationRoad1 << std::endl;
    std::cout << "Road 2 Duration: " << durationRoad2 << std::endl;

    if (potValueRoad1 > potValueRoad2) {
        std::this_thread::sleep_for(std::chrono::milliseconds(1000));

        std::this_thread::sleep_for(std::chrono::milliseconds(2000));

        std::this_thread::sleep_for(std::chrono::milliseconds(durationRoad1));
    } else {
        std::this_thread::sleep_for(std::chrono::milliseconds(1000));

        std::this_thread::sleep_for(std::chrono::milliseconds(2000));

        std::this_thread::sleep_for(std::chrono::milliseconds(durationRoad2));
    }

    std::cout << "Simulating digitalWrite for turning off all LEDs" << std::endl;
}

int TrafficController::map(int x, int in_min, int in_max, int out_min, int out_max) {
    return static_cast<int>(std::round((x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min));
}

int TrafficController::mapWrapper(int x, int in_min, int in_max, int out_min, int out_max) {
    return map(x, in_min, in_max, out_min, out_max);
}
```

Defect Test cases:

```
TEST_F(TrafficControllerTest, IncorrectLoopEqualPotValues) {
    int potValueRoad1 = 512;
    int potValueRoad2 = 512;
    EXPECT_NE(controller.mapWrapper(potValueRoad1, 0, 1023, 5000, 15000),
              controller.mapWrapper(potValueRoad2, 0, 1023, 5000, 15000));
}

TEST_F(TrafficControllerTest, MapWrapperNegativeInput) {
    int result = controller.mapWrapper(-10, 0, 100, 0, 200);
    EXPECT_LT(result, 0);
}
TEST_F(TrafficControllerTest, IncorrectLoopLEDOff) {
    int potValueRoad1 = 700;
    int potValueRoad2 = 300;
    EXPECT_NE(controller.mapWrapper(potValueRoad1, 0, 1023, 5000, 15000),
              controller.mapWrapper(potValueRoad2, 0, 1023, 5000, 15000));
}
TEST_F(TrafficControllerTest, IncorrectMapWrapperBehavior) {
    int potValue = 750;
    int in_min = 500;
    int in_max = 1000;
    int out_min = 2000;
    int out_max = 5000;

    int incorrectResult = controller.mapWrapper(potValue, in_min, in_max, out_max, out_min);

    EXPECT_NE(incorrectResult, controller.mapWrapper(potValue, in_min, in_max, out_min, out_max));
}
TEST_F(TrafficControllerTest, IncorrectMapWrapperOutOfBounds) {
    int potValue = 1100;
    int in_min = 0;
    int in_max = 1000;
    int out_min = 2000;
    int out_max = 5000;

    EXPECT_LT(controller.mapWrapper(potValue, in_min, in_max, out_min, out_max), out_min);
}
```

```
amr@amr-Major-X10:~/Documents/ESE_pro$ ./TrafficControllerTest
bash: ./TrafficControllerTest: No such file or directory
amr@amr-Major-X10:~/Documents/ESE_pro$ g++ -c TrafficController.cpp -o TrafficController.o
amr@amr-Major-X10:~/Documents/ESE_pro$ g++ -c TrafficControllerTest.cpp -o TrafficControllerTest.o -lgtest -lgtest_main -pthread
amr@amr-Major-X10:~/Documents/ESE_pro$ g++ -o TrafficControllerTest TrafficController.o TrafficControllerTest.o -lgtest -lgtest_main -pthread
amr@amr-Major-X10:~/Documents/ESE_pro$ ./TrafficControllerTest
[==========] Running 12 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 12 tests from TrafficControllerTest
[ RUN      ] TrafficControllerTest.MapFunction
[       OK ] TrafficControllerTest.MapFunction (0 ms)
[ RUN      ] TrafficControllerTest.LoopFunction
[       OK ] TrafficControllerTest.LoopFunction (0 ms)
[ RUN      ] TrafficControllerTest.SetupFunction
Simulating pinMode
[       OK ] TrafficControllerTest.SetupFunction (0 ms)
[ RUN      ] TrafficControllerTest.MapWrapperFunction
[       OK ] TrafficControllerTest.MapWrapperFunction (0 ms)
[ RUN      ] TrafficControllerTest.LoopFunctionWithZeroDuration
Road 1 Duration: 5000
Road 2 Duration: 5000
Simulating digitalWrite for turning off all LEDs
[       OK ] TrafficControllerTest.LoopFunctionWithZeroDuration (8000 ms)
[ RUN      ] TrafficControllerTest.LoopFunctionWithEqualPotValues
Road 1 Duration: 10004
Road 2 Duration: 10004
Simulating digitalWrite for turning off all LEDs
[       OK ] TrafficControllerTest.LoopFunctionWithEqualPotValues (13004 ms)
[ RUN      ] TrafficControllerTest.LoopFunctionWithPot1GreaterThanPot2
Road 1 Duration: 12820
Road 2 Duration: 8910
Simulating digitalWrite for turning off all LEDs
[       OK ] TrafficControllerTest.LoopFunctionWithPot1GreaterThanPot2 (15820 ms)
[ RUN      ] TrafficControllerTest.IncorrectLoopEqualPotValues
TrafficControllerTest.cpp:68: Failure
Expected: (controller.mapWrapper(potValueRoad1, 0, 1023, 5000, 15000)) != (controller.mapWrapper(potValueRoad2, 0, 1023, 5000, 15000)), actual: 10004 vs 10004
[  FAILED  ] TrafficControllerTest.IncorrectLoopEqualPotValues (0 ms)
[ RUN      ] TrafficControllerTest.MapWrapperNegativeInput
[       OK ] TrafficControllerTest.MapWrapperNegativeInput (0 ms)
[ RUN      ] TrafficControllerTest.IncorrectLoopLEDOff
[       OK ] TrafficControllerTest.IncorrectLoopLEDOff (0 ms)
[ RUN      ] TrafficControllerTest.IncorrectMapWrapperBehavior
TrafficControllerTest.cpp:98: Failure
Expected: (incorrectResult) != (controller.mapWrapper(potValue, in_min, in_max, out_min, out_max)), actual: 3500 vs 3500
[  FAILED  ] TrafficControllerTest.IncorrectMapWrapperBehavior (0 ms)
[ RUN      ] TrafficControllerTest.IncorrectMapWrapperOutOfBounds
TrafficControllerTest.cpp:109: Failure
Expected: (controller.mapWrapper(potValue, in_min, in_max, out_min, out_max)) < (out_min), actual: 5300 vs 2000
[  FAILED  ] TrafficControllerTest.IncorrectMapWrapperOutOfBounds (0 ms)
[----------] 12 tests from TrafficControllerTest (36826 ms total)

[----------] Global test environment tear-down
[==========] 12 tests from 1 test suite ran. (36826 ms total)
[  PASSED  ] 9 tests.
[  FAILED  ] 3 tests, listed below:
[  FAILED  ] TrafficControllerTest.IncorrectLoopEqualPotValues
[  FAILED  ] TrafficControllerTest.IncorrectMapWrapperBehavior
[  FAILED  ] TrafficControllerTest.IncorrectMapWrapperOutOfBounds

 3 FAILED TESTS
Some tests failed.
amr@amr-Major-X10:~/Documents/ESE_pro$
```

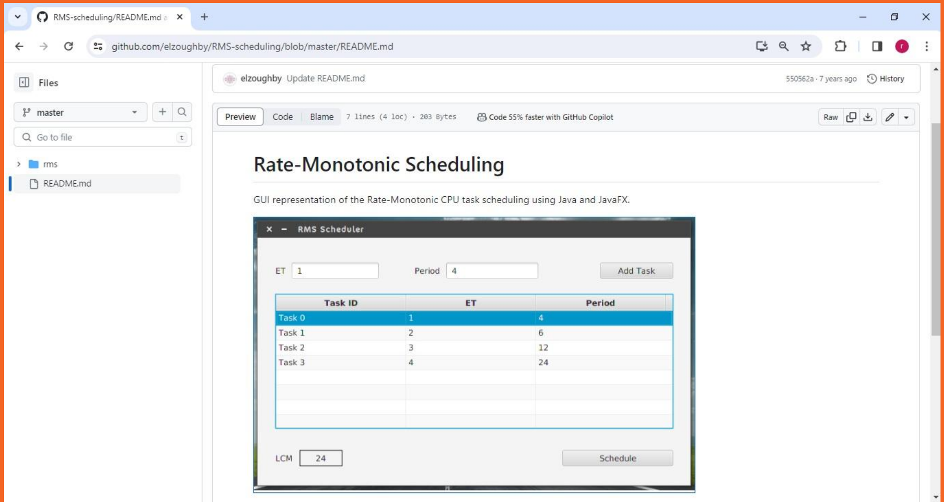
Inspection report

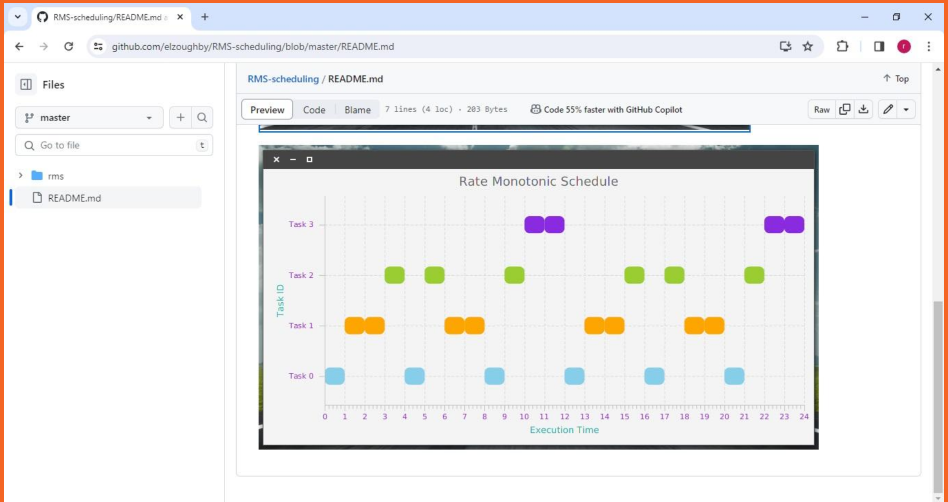


Scheduling

- JavaFX
- Tkinter
- Python-RTS
- Creating EDF and RMS
- Analysis

JavaFX


RMS-scheduling/README.md
github.com/elzoughby/RMS-scheduling/blob/master/README.md
Files
master
Go to file
rms
README.md
elzoughby Update README.md
550562a · 7 years ago
History
Preview
Code
Blame
7 lines (4 loc) · 203 Bytes
Code 55% faster with GitHub Copilot
Raw
Rate-Monotonic Scheduling
GUI representation of the Rate-Monotonic CPU task scheduling using Java and JavaFX.
RMS Scheduler
ET
1
Period
4
Add Task
Task ID
ET
Period
Task 0
1
4
Task 1
2
6
Task 2
3
12
Task 3
4
24
LCM
24
Schedule


RMS-scheduling/README.md
github.com/elzoughby/RMS-scheduling/blob/master/README.md
Files
master
Go to file
rms
README.md
RMS-scheduling / README.md
Top
Preview
Code
Blame
7 lines (4 loc) · 203 Bytes
Code 55% faster with GitHub Copilot
Raw
Rate Monotonic Schedule
Task 3
Task 2
Task 1
Task 0
Task ID
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Execution Time


JavaFX
Documentation
Community
Testimonials
Highlights
JavaFX
JavaFX is an open source, next generation client application platform for desktop, mobile and embedded systems built on Java. It is a collaborative effort by many individuals and companies with the goal of producing a modern, efficient, and fully featured toolkit for developing rich client applications.

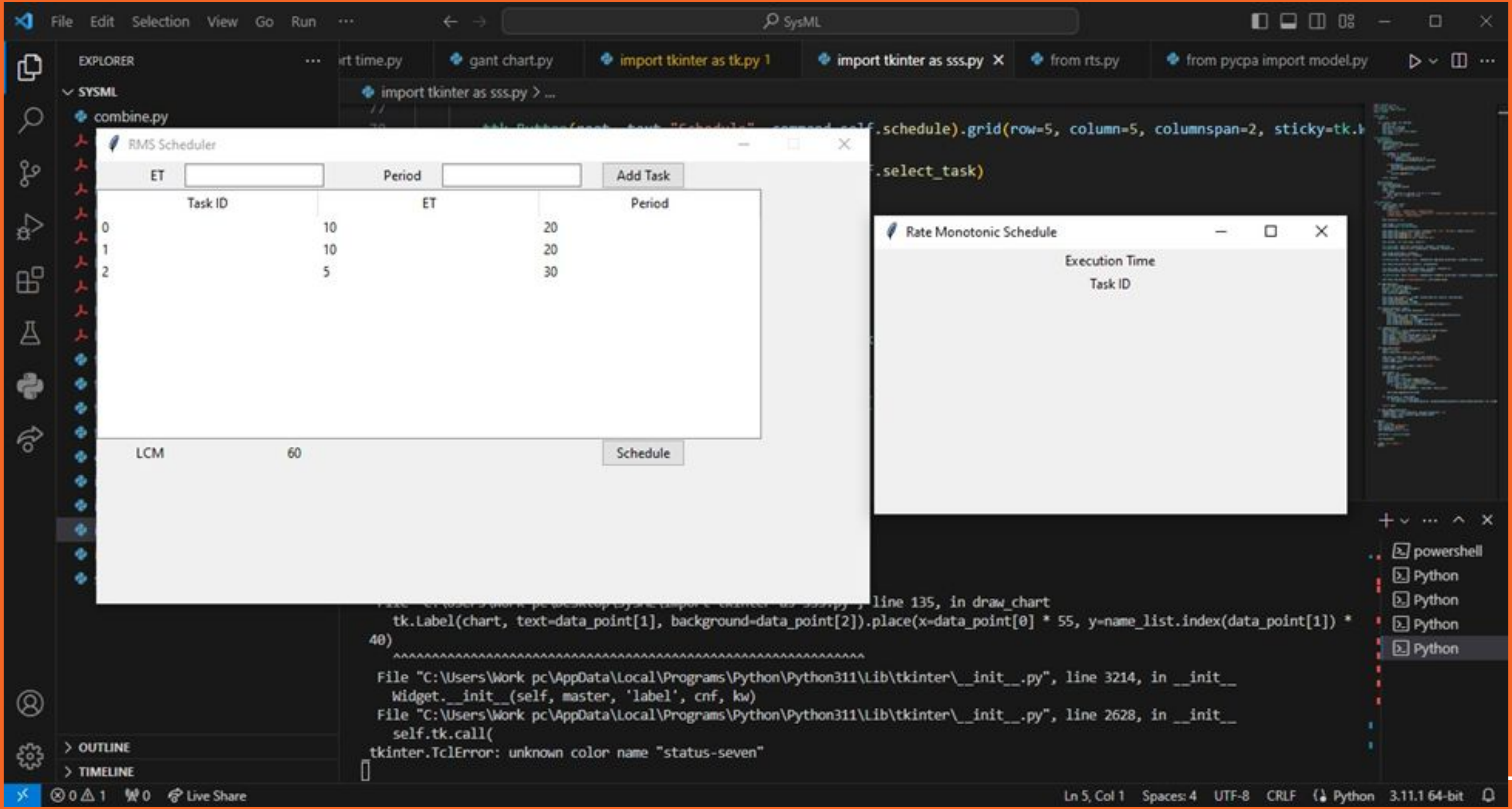
Python with Tkinter

```python
import tkinter as tk
from tkinter import ttk
from functools import partial
import random

class Task:
    count = 0

    def __init__(self, eT, period):
        self.eT = eT
        self.period = period
        self.id = Task.count
        self.name = f"Task {Task.count}"
        Task.count += 1

class Scheduler:
    @staticmethod
    def schedule(taskList):
        lcm = Scheduler.calcLCM(taskList)
        waitingList = []
        outList = []

        for timeUnit in range(lcm):
            for task in taskList:
                if timeUnit % task.period == 0:
                    waitingList.extend([task] * task.eT)

            if waitingList:
                waitingList.sort(key=lambda x: x.period)
                outList.append(waitingList.pop(0))
            else:
                outList.append(None)

        return outList

    @staticmethod
    def calcLCM(taskList):
        lcm = taskList[0].period
        flag = True
        while flag:
            flag = any(lcm % x.period != 0 for x in taskList)
            lcm = lcm + 1 if flag else lcm
        return lcm

class Controller:
    def __init__(self, root):
        self.taskList = []
        self.style = [
            "status-one", "status-two", "status-three",
            "status-four", "status-five", "status-six", "status-seven", "status-
eight", "status-nine", "status-ten",
            "status-eleven", "status-twelve"
        ]
        self.styleList = []
```

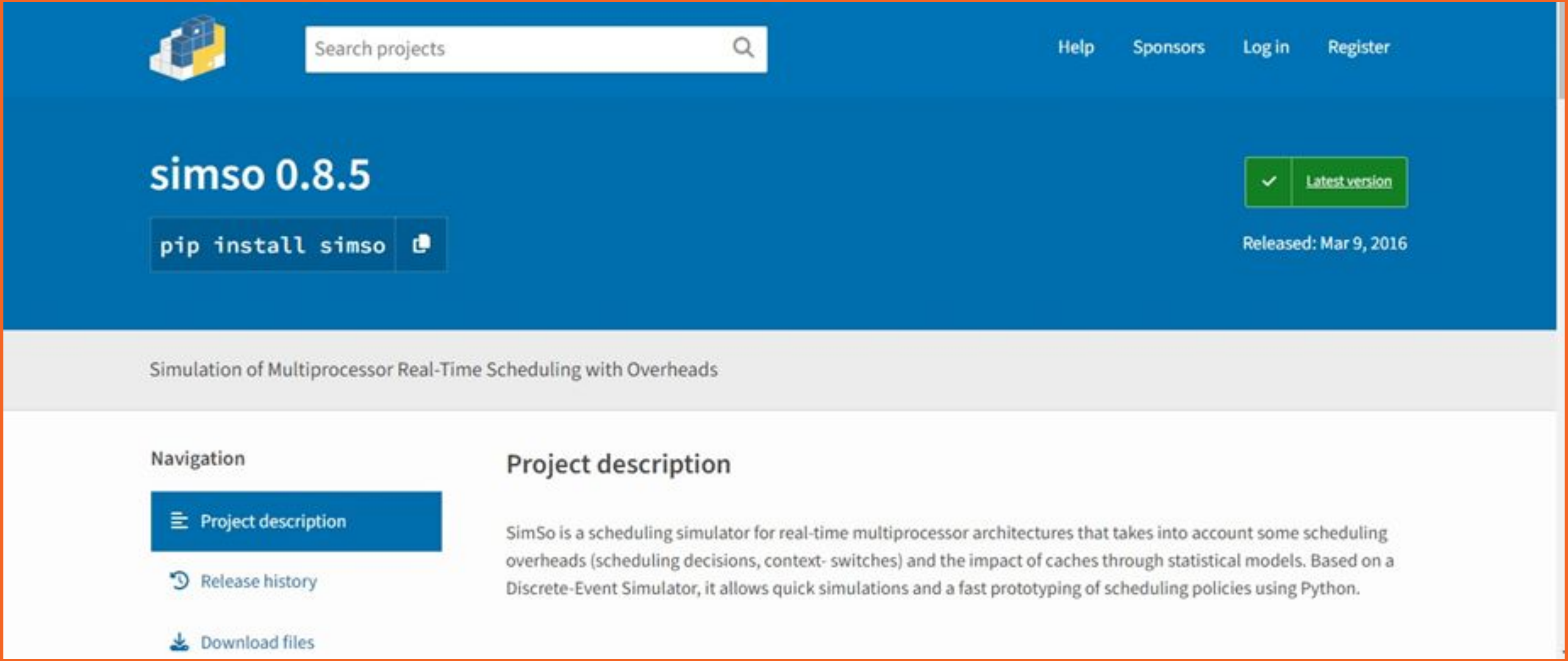
```python
        self.txtET = ttk.Entry(root)
        self.txtPeriod = ttk.Entry(root)

        self.task_tree = ttk.Treeview(root, columns=("ID", "ET", "Period"),
show="headings")
        self.task_tree.heading("ID", text="Task ID")
        self.task_tree.heading("ET", text="ET")
        self.task_tree.heading("Period", text="Period")

        self.lblLCM = ttk.Label(root, text="0")

        ttk.Label(root, text="ET").grid(row=1, column=1, sticky=tk.E)
        ttk.Label(root, text="Period").grid(row=1, column=3, sticky=tk.E)

        self.txtET.grid(row=1, column=2)
        self.txtPeriod.grid(row=1, column=4)

        ttk.Button(root, text="Add Task", command=self.add_task).grid(row=1,
column=5, sticky=tk.W)

        self.task_tree.grid(row=3, column=1, columnspan=6)

        ttk.Label(root, text="LCM").grid(row=5, column=1, sticky=tk.E)
        self.lblLCM.grid(row=5, column=2, columnspan=2)

        ttk.Button(root, text="Schedule", command=self.schedule).grid(row=5,
column=5, columnspan=2, sticky=tk.W)

        self.task_tree.bind("<<TreeviewSelect>>", self.select_task)

    def add_task(self):
        eT = int(self.txtET.get())
        period = int(self.txtPeriod.get())
        task = Task(eT, period)
        self.taskList.append(task)

        self.task_tree.insert("", tk.END, values=(task.id, task.eT,
task.period))
        self.txtET.delete(0, tk.END)
        self.txtPeriod.delete(0, tk.END)
        self.lblLCM.config(text=str(Scheduler.calcLCM(self.taskList)))

    def select_task(self, event):
        selection = self.task_tree.selection()
        if selection:
            selected_task = self.taskList[int(self.task_tree.index(selection))]
            self.txtET.delete(0, tk.END)
            self.txtET.insert(0, str(selected_task.eT))
            self.txtPeriod.delete(0, tk.END)
            self.txtPeriod.insert(0, str(selected_task.period))

    def schedule(self):
        self.styleList = random.sample(self.style, len(self.style))
```

```python
        chart = self.draw_chart()
        chart_width = int(self.lblLCM.cget("text")) * 55
        chart_height = len(self.taskList) * 20 * 2 + 100
        chart.geometry(f"{chart_width}x{chart_height}")
        chart.title("Rate Monotonic Schedule")
        chart.mainloop()

    def draw_chart(self):
        chart = tk.Tk()
        chart.title("Rate Monotonic Schedule")

        name_list = [task.name for task in self.taskList]
        x_axis_label = tk.Label(chart, text="Execution Time")
        x_axis_label.pack()

        y_axis_label = tk.Label(chart, text="Task ID")
        y_axis_label.pack()

        chart_data = []
        for task in self.taskList:
            series_data = []
            style_class = self.get_random_style()
            result_list = Scheduler.schedule([task])
            for i, result_task in enumerate(result_list):
                if task == result_task:
                    series_data.append((i, task.name, style_class))

            chart_data.append(series_data)

        for series_data in chart_data:
            for data_point in series_data:
                tk.Label(chart, text=data_point[1],
background=data_point[2]).place(x=data_point[0] * 55,
y=name_list.index(data_point[1]) * 40)

        return chart

    def get_random_style(self):
        random_index = random.randint(0, len(self.styleList) - 1)
        random_style = self.styleList.pop(random_index)
        return random_style

def main():
    root = tk.Tk()
    root.title("RMS Scheduler")
    root.geometry("700x400")
    root.resizable(False, False)

    controller = Controller(root)

    root.mainloop()

if __name__ == "__main__":
    main()
```

Python with Tkinter



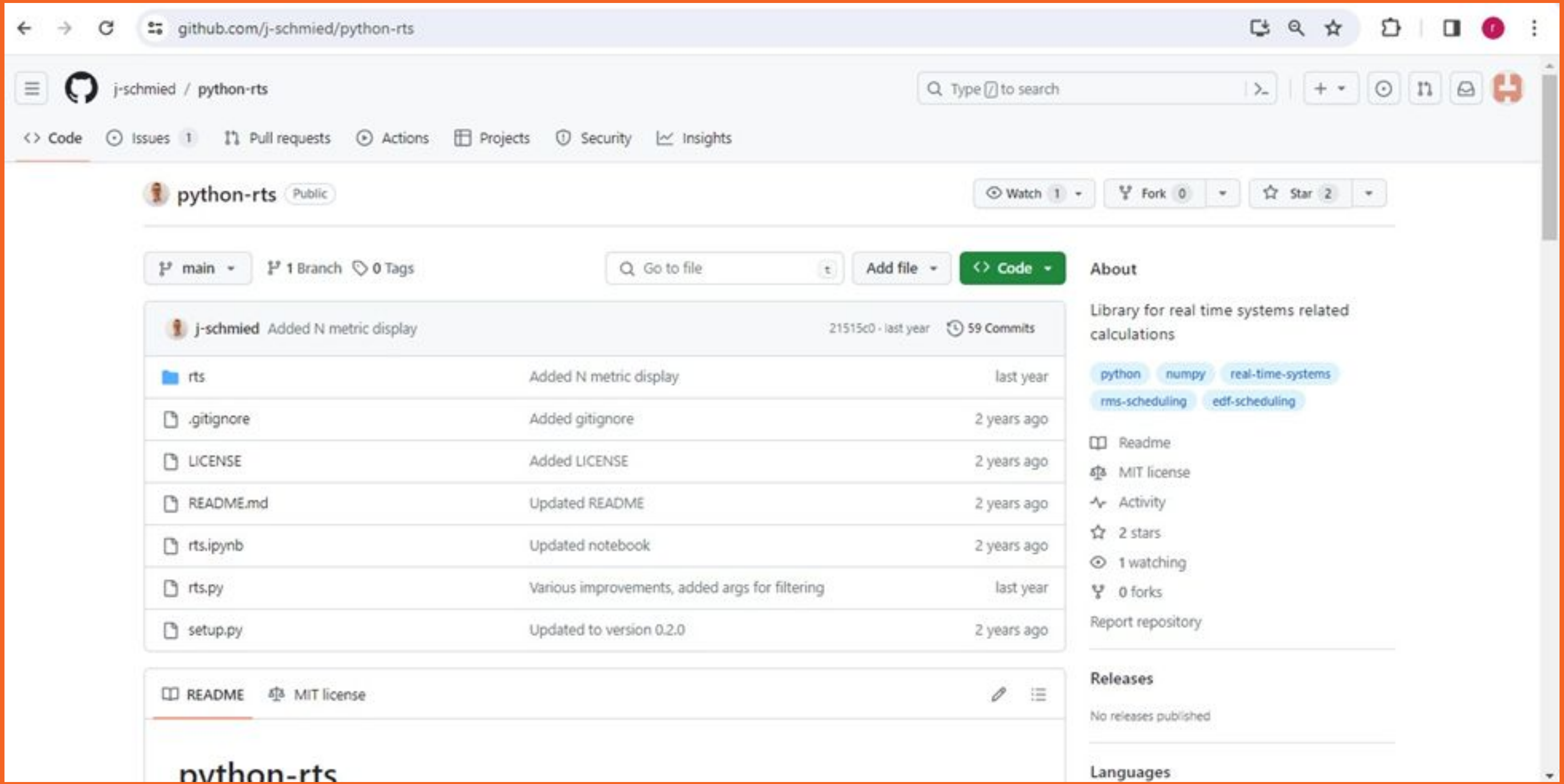
SIMSO


Search projects
Help
Sponsors
Log in
Register
simso 0.8.5
pip install simso
Latest version
Released: Mar 9, 2016
Simulation of Multiprocessor Real-Time Scheduling with Overheads
Navigation
Project description
Release history
Download files
Project description
SimSo is a scheduling simulator for real-time multiprocessor architectures that takes into account some scheduling overheads (scheduling decisions, context- switches) and the impact of caches through statistical models. Based on a Discrete-Event Simulator, it allows quick simulations and a fast prototyping of scheduling policies using Python.

SIMSO

```python
from simso.schedulers import RMS, EDF
from simso.core import Model, ProcEvent, Task

def simulate(scheduler):
    model = Model()

    # Add processors
    proc = ProcEvent(model, scheduler)

    model.scheduler = scheduler
    model.procs = [proc]
    model.add_tasks(tasks)
    model.run_model()

    # Print scheduling information
    for task in model.results.tasks:
        print(f"{task.name} - {task.starting_date} - {task.ending_date}")

# Instantiate scheduler classes
edf_scheduler = EDF()
rm_scheduler = RMS()

# Simulate with Earliest Deadline First (EDF)
print("EDF Schedule:")
simulate(edf_scheduler)

# Simulate with Rate Monotonic (RMS)
print("\nRate Monotonic (RMS) Schedule:")
simulate(rm_scheduler)
```

Python-RTS


github.com/j-schmied/python-rts
j-schmied / python-rts
Type / to search
Code
Issues 1
Pull requests
Actions
Projects
Security
Insights
python-rts
Public
Watch 1
Fork 0
Star 2
main
1 Branch
0 Tags
Go to file
Add file
Code
j-schmied Added N metric display
21515c0 · last year
59 Commits
rts
Added N metric display
last year
.gitignore
Added gitignore
2 years ago
LICENSE
Added LICENSE
2 years ago
README.md
Updated README
2 years ago
rts.ipynb
Updated notebook
2 years ago
rts.py
Various improvements, added args for filtering
last year
setup.py
Updated to version 0.2.0
2 years ago
README
MIT license
python-rts
About
Library for real time systems related calculations
python
numpy
real-time-systems
rms-scheduling
edf-scheduling
Readme
MIT license
Activity
2 stars
1 watching
0 forks
Report repository
Releases
No releases published
Languages

Python EDF & RMS

```python
import time
import random
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import pandas as pd

class Task:
    def __init__(self, name, period, deadline, execution_time):
        self.name = name
        self.period = period
        self.deadline = deadline
        self.next_release_time = time.time() + period
        self.execution_time = execution_time
        self.execution_times = []

    def execute(self, current_time):
        self.execution_times.append(current_time)
        current_time_seconds = int(current_time)  # Extracting the integer part
(seconds)
        last_three_digits = current_time_seconds % 1000  # Extracting the last
three digits
        print(f"{self.name} is executing at time {last_three_digits}")

    def update_next_release_time(self):
        self.next_release_time += self.period

def visualize_execution(tasks, num_periods):
    replicated_tasks = []
    for i in range(1, num_periods):
        replicated_tasks.extend(
            {
                "Task": task.name,
                "Start": task.execution_times[-1],
                "Finish": task.execution_times[-1] + task.period,
            }
            for task in tasks
        )

    df = pd.DataFrame(replicated_tasks)
    bar_height = 0.1

    fig, ax = plt.subplots(figsize=(10, bar_height * len(df["Task"].unique())))

    ax.set_yticks([i * bar_height + bar_height / 2 for i in
range(len(df["Task"].unique()))])
    ax.set_yticklabels(df["Task"].unique())
```

```python
    for i, task in enumerate(df["Task"].unique()):
        task_df = df[df["Task"] == task]
        ax.broken_barh(
            [(start, finish - start) for start, finish in zip(task_df["Start"],
task_df["Finish"])],
            (i * bar_height, bar_height),
            facecolors=["blue", "orange", "green"],
        )

    ax.set_xlabel("Time")
    ax.set_title(f"Scheduling Visualization ({num_periods} Periods)")

    plt.show()

def edf_scheduler(tasks, simulation_time):
    start_time = time.time()

    while time.time() - start_time < simulation_time:
        current_time = time.time()
        ready_tasks = [task for task in tasks if current_time >=
task.next_release_time]

        if ready_tasks:
            earliest_deadline_task = min(ready_tasks, key=lambda task:
task.deadline)
            earliest_deadline_task.execute(current_time)
            earliest_deadline_task.update_next_release_time()

        for task in tasks:
            if task.next_release_time <= current_time:
                task.execute(current_time)
                task.update_next_release_time()

        time.sleep(1)

    visualize_execution(tasks, int(simulation_time / min(task.period for task in
tasks)))

def rms_scheduler(tasks, simulation_time):
    start_time = time.time()

    while time.time() - start_time < simulation_time:
        current_time = time.time()
        ready_tasks = [task for task in tasks if current_time >=
task.next_release_time]

        if ready_tasks:
```

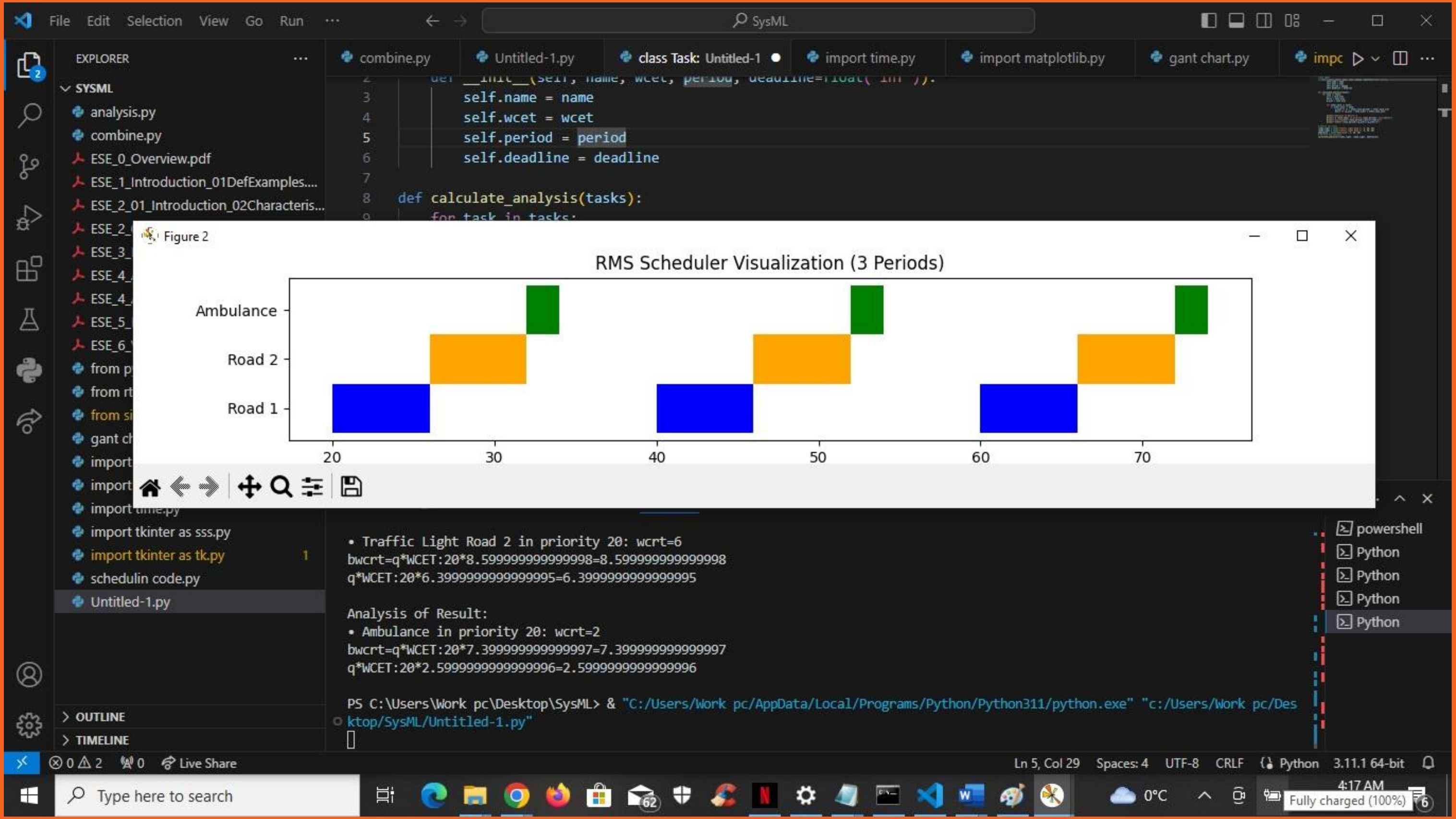
```python
            highest_priority_task = min(ready_tasks, key=lambda task:
task.period)
            highest_priority_task.execute(current_time)
            highest_priority_task.update_next_release_time()

        for task in tasks:
            if task.next_release_time <= current_time:
                task.execute(current_time)
                task.update_next_release_time()

        time.sleep(1)

    visualize_execution(tasks, int(simulation_time / min(task.period for task in
tasks)))

def main_edf():
    road1_light = Task("Traffic Light Road 1", 20, 20, 8)
    road2_light = Task("Traffic Light Road 2", 20, 20, 8)
    ambulance = Task("Ambulance", 20, 20, 2)

    tasks = [road1_light, road2_light, ambulance]

    edf_scheduler(tasks, simulation_time=300)

def main_rms():
    road1_light = Task("Traffic Light Road 1", 20, 20, 5)
    road2_light = Task("Traffic Light Road 2", 40, 20, 5)
    ambulance = Task("Ambulance", 80, 10, 5)

    tasks = [road1_light, road2_light, ambulance]

    rms_scheduler(tasks, simulation_time=300)

if __name__ == "__main__":
    main_edf()
    # Uncomment the line below to run RMS scheduler
    # main_rms()
```

Python EDF & RMS

Python EDF & RMS

```python
class Task:
    def __init__(self, name, wcet, period, deadline=float('inf')):
        self.name = name
        self.wcet = wcet
        self.period = period
        self.deadline = deadline

def calculate_analysis(tasks):
    for task in tasks:
        wcrt = task.wcet
        bwcrt = task.wcet
        q_wcet = task.wcet

        for other_task in tasks:
            if other_task != task:
                q_wcet += (1 / other_task.period) * other_task.wcet
                bwcrt += (q_wcet - task.wcet) * other_task.wcet

        print(f"Analysis of Result:")
        print(f"• {task.name} in priority {task.period}: wcrt={wcrt}")
        print(f"bwcrt=q*WCET:{task.period}*{bwcrt}={bwcrt}")
        print(f"q*WCET:{task.period}*{q_wcet}={q_wcet}\n")

# Define the tasks
road1_light = Task("Traffic Light Road 1", 6, 20, 20)
road2_light = Task("Traffic Light Road 2", 6, 20, 20)
ambulance = Task("Ambulance", 2, 20, 20)
# Perform the analysis
calculate_analysis([road1_light, road2_light, ambulance])
```

Python EDF & RMS



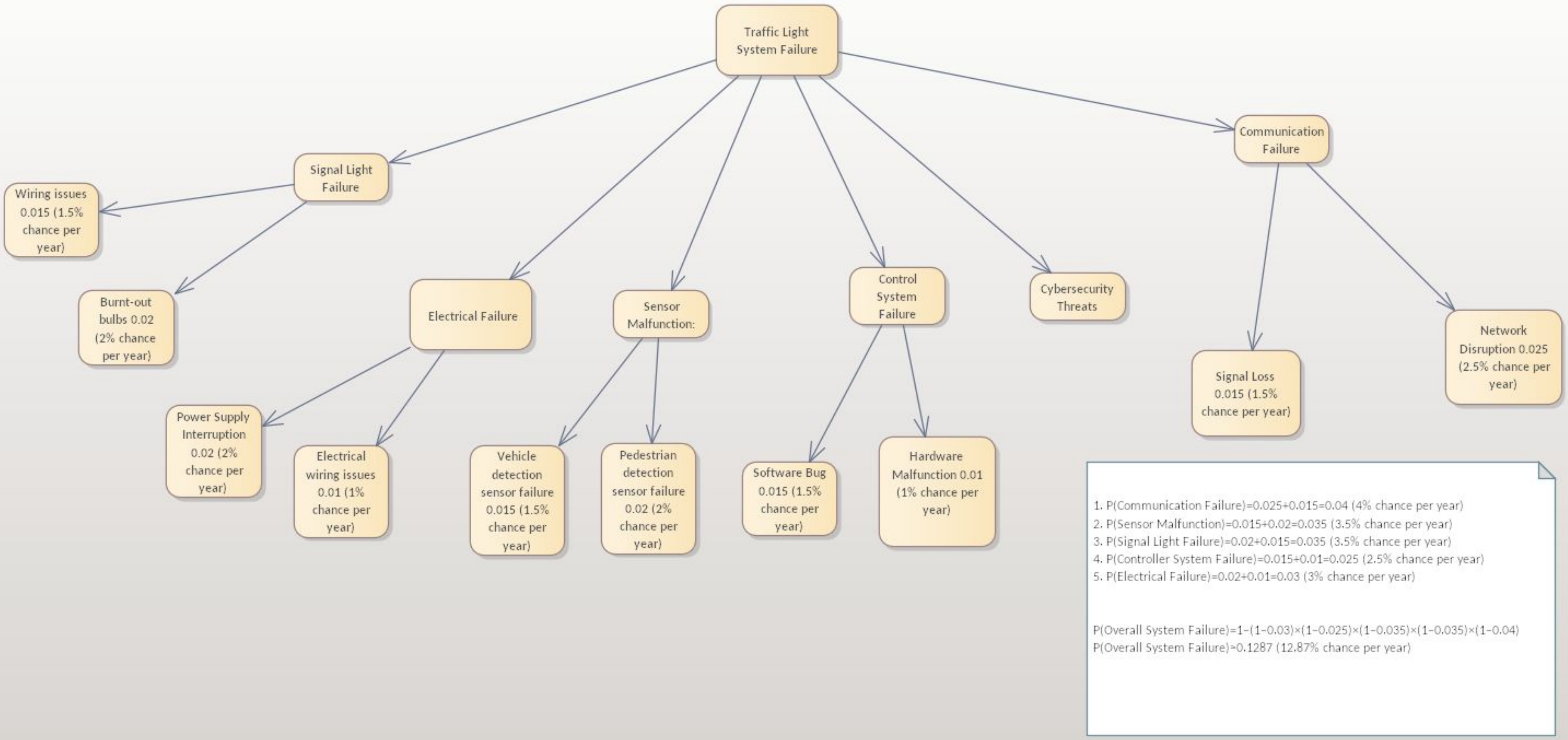
Python EDF & RMS Analysis

```python
class Task:
    def __init__(self, name, wcet, period, deadline=float('inf')):
        self.name = name
        self.wcet = wcet
        self.period = period
        self.deadline = deadline

def calculate_analysis(tasks):
    for task in tasks:
        wcrt = task.wcet
        bwcrt = task.wcet
        q_wcet = task.wcet

        for other_task in tasks:
            if other_task != task:
                q_wcet += (1 / other_task.period) * other_task.wcet
                bwcrt += (q_wcet - task.wcet) * other_task.wcet

        print(f"Analysis of Result:")
        print(f"• {task.name} in priority {task.period}: wcrt={wcrt}")
        print(f"bwcrt=q*WCET:{task.period}*{bwcrt}={bwcrt}")
        print(f"q*WCET:{task.period}*{q_wcet}={q_wcet}\n")

# Define the tasks
road1_light = Task("Traffic Light Road 1", 6, 20, 20)
road2_light = Task("Traffic Light Road 2", 6, 20, 20)
ambulance = Task("Ambulance", 2, 20, 20)  # Assuming a WCET of 2 for the
ambulance

# Perform the analysis
calculate_analysis([road1_light, road2_light, ambulance])
```

Python EDF & RMS Analysis

```
PROBLEMS 2   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

ktop/SysML/analysis.py"
Analysis of Result:
• Traffic Light Road 1 in priority 20: wcrt=6
bwcrt=q*WCET:20*8.599999999999998=8.599999999999998
q*WCET:20*6.3999999999999995=6.3999999999999995

• Traffic Light Road 2 in priority 20: wcrt=6
bwcrt=q*WCET:20*8.599999999999998=8.599999999999998
q*WCET:20*6.3999999999999995=6.3999999999999995

Analysis of Result:
• Ambulance in priority 20: wcrt=2
bwcrt=q*WCET:20*7.399999999999997=7.399999999999997
q*WCET:20*2.5999999999999996=2.5999999999999996
```

Traffic Light Hazards with probabilities



Traffic Light System Failure
Signal Light Failure
Wiring issues 0.015 (1.5% chance per year)
Burnt-out bulbs 0.02 (2% chance per year)
Electrical Failure
Power Supply Interruption 0.02 (2% chance per year)
Electrical wiring issues 0.01 (1% chance per year)
Sensor Malfunction:
Vehicle detection sensor failure 0.015 (1.5% chance per year)
Pedestrian detection sensor failure 0.02 (2% chance per year)
Control System Failure
Software Bug 0.015 (1.5% chance per year)
Hardware Malfunction 0.01 (1% chance per year)
Cybersecurity Threats
Communication Failure
Signal Loss 0.015 (1.5% chance per year)
Network Disruption 0.025 (2.5% chance per year)
1. P(Communication Failure)=0.025+0.015=0.04 (4% chance per year)
2. P(Sensor Malfunction)=0.015+0.02=0.035 (3.5% chance per year)
3. P(Signal Light Failure)=0.02+0.015=0.035 (3.5% chance per year)
4. P(Controller System Failure)=0.015+0.01=0.025 (2.5% chance per year)
5. P(Electrical Failure)=0.02+0.01=0.03 (3% chance per year)
P(Overall System Failure)=1−(1−0.03)×(1−0.025)×(1−0.035)×(1−0.035)×(1−0.04)
P(Overall System Failure)≈0.1287 (12.87% chance per year)

Hazard Handling at Design Level

- *superfluity: Backup power and communication systems.*
- *Sensor Calibration: Regular maintenance and calibration.*
- *Software Validation: Rigorous testing and validation.*
- *Hardware Redundancy: Duplicate critical components.*

Hazard Handling at Implementation Level

- *Regular Maintenance: Routine checks and maintenance.*
- *Remote Monitoring: Systems for remote monitoring.*
- *Security Measures: Cybersecurity protocols and firewalls.*

Design and Development Process

We identified and refined at least 10 requirements, defined the context and use cases, and visualized the system's architecture with block diagrams and sequence diagrams.

Implementation and Testing

Our work extends to the implementation level, showcasing the mapping to prototype hardware, including tinkercad platform.

We've executed 5 unit tests, defined and executed 5 defect tests, and conducted 3 component/interface tests. Our implementation has followed a test-driven development approach, ensuring reliability and functionality.

Scheduling and Performance Analysis

Breakdown of requirements into scheduling, utilizing pyCPA for computation time analysis, and ensuring the schedulability of implemented components for optimal system performance.

Thank you